File Copy

BRL MR 2053

BRL

MEMORANDUM REPORT NO. 2053

FUNCTION APPROXIMATION ROUTINES FOR BRLESC - A SURVEY

by

John D. Wortman

August 1970

U.S. ARMY ABERDEEN RESEARCH AND DEVELOPMENT CENTER
BALLISTIC RESEARCH LABORATORIES
ABERDEEN PROVING GROUND, MARYLAND

Destroy this report when it is no longer needed.
Do not return it to the originator.

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

B A L L I S T I C R E S E A R C H L A B O R A T O R I E S

MEMORANDUM REPORT NO. 2053

AUGUST 1970

FUNCTION APPROXIMATION ROUTINES FOR BRLESC - A SURVEY

John D. Wortman

Applied Mathematics Division

RDT&E Project No. 1T061102A14B

A B E R D E E N P R O V I N G G R O U N D, M A R Y L A N D

B A L L I S T I C R E S E A R C H L A B O R A T O R I E S

MEMORANDUM REPORT NO. 2053

JDWortman/bj
Aberdeen Proving Ground, Md.
August 1970

FUNCTION APPROXIMATION ROUTINES FOR BRLESC - A SURVEY

ABSTRACT

This report is a survey of the function approximation routines available for the BRLESC computer at Aberdeen Proving Ground as of April 1969. It includes a description of each routine and some general discussion.

TABLE OF CONTENTS

I. INTRODUCTION

The approximation of a function by another function is a frequent computing problem. The primary purpose of this report is to list the readily available routines of this type for the BRLESC[1*] at Aberdeen Proving Ground, as of April 1969. The majority of these routines, and descriptions of them, are available through the Computer Support Division, Systems Programming (Building 328, Room 213). Anyone who is considering Curve fitting is strongly urged to examine the existing programs before attempting to code his own.

In nearly all of these routines the original function is represented by a table of N data points, (X_i,y_i), i=1,2,...,N, where y_i is the dependent variable (the function value we wish to approximate) and X_i is either a single dependent variable, x_i, or a vector of variables, $X_i = (x_{i1},x_{i2},\ldots,X_{iN})^T$. The immediate goal of most of the routines is to find the "best" values for M parameters $A = (a_1,a_2,\ldots,a_M)$ in an equation F(X,A), the approximating function, which has been chosen to represent the table of data.

In all these routines, the parameters, A, are chosen so as to satisfy some prescribed constraints or to optimize some measure of the goodness of the approximation. Some of the routines do both. The most popular measure is the variance of residuals (the sum of squares of the residuals) over the set of data being fitted. Fitting with this measure is called the "method of least squares". Most numerical analysis textbooks have some discussion of the principle of least squares (e.g. Hildebrand[4]). Most of the fitting programs for BRLESC are based on this principle which states that the "best" approximation is that which minimizes the sum of squares of the residuals.

The only approximation routines we consider are for interpolation or curve fitting. We will not discuss the approximations used for any of the standard functions (SIN, EXP, etc.), nor will we consider any

*References are listed on pages 40 and 41.

integration routines. Undoubtedly, some useful programs were overlooked, other good BRLESC approximation routines may be completed in the near future, and FORTRAN[2] versions of the most useful FORAST[3] routines will be made. (The MULTIPLE REGRESSION and NONLINEAR LEAST SQUARES are now available in FORTRAN.)

In this report we first list the FORTRAN and FORAST function approximation routines which were located and give a brief description of them. We then have some general discussion of these programs. Finally, we discuss each routine in a little more detail, give references, point out some difficulties, and indicate some advantages and disadvantages of some of the routines.

Instructions for using many of the routines are included in the appendix.

II. LIST OF ROUTINES

A. Standard FORTRAN and (FORAST) Subroutines

Interpolation Subroutine

DVDINT (D.D.IN)

Least Squares Subroutines

MATINV, FNEQS (S.N.E., F.N.E., etc.)
GENLSQ (G.L.SQ)
POLYLS (P.L.SQ)

Function Minimizing Subroutines

FNMIN (FN.MIN)
FDMIN (FD.MIN)

B. FORAST Fitting Programs

LEAST SQUARES PROGRAM
MULTIPLE REGRESSION (FORTRAN program available)
NONLINEAR LEAST SQUARES (FORTRAN program available)
PIECEWISE QUARTIC FIT
CUBIC SPLINE
LEAST SQUARES CUBIC SPLINE
POLYGONAL CURVE

C. Other FORTRAN Routines

CONSTRAINED NONLINEAR LEAST SQUARES (Program)
NONLINEAR LEAST SQUARES FOR CORRELATED DATA
NLPROG (Subroutine to minimize constrained functions)

The "standard" FORTRAN subroutines in the above list are available on cards from Systems Programming, Computer Support Division. Each of these subroutines is equivalent to some FORAST Subroutine on the FORAST compiler tape. Three of the FORAST programs, LEAST SQUARES PROGRAM, MULTIPLE REGRESSION, and NONLINEAR LEAST SQUARES, are also available from Systems Programming. Using routines from this source has several advantages: The routines are available in a standard usable form. Descriptions are available. The routines are completely checked out. And advice and assistance are available if needed. The other FORAST programs and the "other" FORTRAN routines have special features which may compensate for their defects in availability, standardization, etc.

We have classified these routines as interpolating routines, fitting routines, least squares routines, and function minimizing routines. Such a division must be arbitrary. We have elected to call the method "interpolation" if the principal result is the value of the approximating function at a prescribed value of the independent variable. If the main output of the routine is an equation or the values of parameters in a prescribed equation, we call it a "fitting" routine. Those fitting procedures which use the principle of least squares are referred to as "least squares" routines.

It seems to be general practice to drop the adjective "linear" in discussion of linear least squares and to use the adjective "nonlinear" when referring to least squares procedures which are not necessarily linear in the unknown parameters. This practice has been followed here in naming and discussing the routines. To further complicate these titles, the adjectives "general" and "polynomial" are used to denote the type of terms permitted in <u>linear</u> least squares.

The function minimizing routines are separated from the other routines because the user is free to choose the measure of the goodness of the approximation. The user pays for this freedom with coding and some loss of confidence in the results.

DVDINT (D.D.IN in FORAST)

This standard subroutine uses divided differences to approximate the value of y corresponding to x from a table (x_i,y_i), i=1,2,...,N, ordered on x. The approximating function is an M-th degree polynomial through M + 1 points in the neighborhood of x.

MATINV, FNEQS (S.N.E., F.N.E., etc. in FORAST)

The MATINV subroutine is the standard FORTRAN subroutine for solving a system of linear equations. The FNEQS subroutine may be used to help convert raw data to the normal equations of least squares and the MATINV subroutine may be used to solve this system. In FORAST, the F.N.E. may be used to help form the normal equations, in the form of an augmented upper triangular matrix, and the SY.SNE or SY.INV used to solve the system. One can also use the F.O.MAT to form the full rectangular augmented matrix from the triangular form and use the S.N.E. or MAT.INV. to solve the system.

It is generally better to use one of the next two subroutines for linear least squares.

GENLSQ (G.L.SQ in FORAST)

Starting from a table of data $(y_i,\varphi_1(X_i),\varphi_2(X_i),\ldots,\varphi_M(X_i))$, i=1,2,...,N, this routine carries out a linear least squares fit. It finds $A = (a_1,a_2,\ldots,a_M)$ so that $U(A) = \sum_{i=1}^{N}(y_i - \sum_{j=1}^{M}a_j\varphi_j(X_i))^2$ is a minimum with respect to the a_j's. This routine will also compute some information to help the user analyze the fit: the root-mean-square error, $(U(A)/(N-M))^{\frac{1}{2}}$; the residuals, $R_i = y_i - \sum_{j=1}^{N}a_j\varphi_j(X_i)$; σ_j, the estimates of the error in a_j; and $t_j = a_j/\sigma_j$, the estimates of the significance of a_j.

The FORAST subroutine allows more flexibility in the storage of the input data and permits more flexibility in weighting or eliminating data points.

POLYLS (P.L.SQ in FORAST)

This is simply a special case of GENLSQ (G.L.SQ) for polynomial least squares with one independent variable. The input is N pairs of numbers (x_i, y_i). The program minimizes the equation $U(A) = \sum_{i=1}^{N}(y_i - \sum_{j=1}^{N} a_j x_i^{j-1})^2$. The FORAST program has an option for dropping some of the powers of x.

FNMIN (FN.MIN in FORAST)

This is a standard subroutine for finding the minimum of a function of several variables without using derivatives. The least squares programs we have just discussed find the minimum of $U(A) = (\sum_{i=1}^{N} R_i^2)^{\frac{1}{2}}$, where the R_i are linear in $a_1, a_2, \ldots, a_M$. This minimizing program could be used to minimize U(A) if the R_i are not linear in the a_j's or if some other measure of the goodness of the fit were used. The user must supply programming to evaluate U(A).

FDMIN (FD.MIN in FORAST)

This is another function minimizing routine. It uses the first partial derivatives of the function to be minimized. The user must supply programming which evaluates U(A) and $\partial U(A)/\partial a_j$, $j=1,2,\ldots,M$.

LEAST SQUARES PROGRAM

This complete FORAST program reads control cards that specify the type and form of a linear least squares fit, reads the data, computes the fit, and prints the coefficients, the approximate function value at each point, the corresponding residuals, the root-mean-square error, and the "sigma and t" error indicators.

MULTIPLE REGRESSION PROGRAM

This complete FORAST program is similar to the previous program in set up and control. The very important difference is that the given approximating equation is a listing of possible terms to be included in a linear least squares fit. This routine selects an equation that has only significant terms and then carries out a least squares fit to find the best coefficients for this equation. In addition to output like the LEAST SQUARES PROGRAM, this program prints a running account of the terms added to or dropped from the tentative equation and prints the terms in the final equation.

NONLINEAR LEAST SQUARES

This complete FORAST program is a program for finding the "best" parameters in the least squares sense for an approximating function which is not linear in all the parameters. If the function $F(X,A)$ is used to approximate y, the best approximation in the least squares sense occurs when $U(A) = \sum_{i=1}^{N}(y_i - F(X_i,A))^2$ is a minimum. $F(X,A)$ is approximated with the linear terms of Taylor's series (i.e., $F(X_i,A) = F(X_i,\bar{A}) + \sum_{j=1}^{M}\Delta a_j \partial F_i/\partial a_j$, where $\bar{A}$ is a good initial approximation to A, and $\partial F_i/\partial a_j$ is the partial derivative of $F(X,A)$ evaluated for $A = \bar{A}$ and $X = X_i$). The standard procedure for linear least squares is used to find Δa_j, $j=1,2,\ldots,M$, so that $U(\Delta A) = \sum_{i=1}^{N}(y_i - F(X,\bar{A}) - \sum_{j=1}^{M}\Delta a_j \partial F_i/\partial a_j)^2$ is minimized. Then $\bar{a}_j$, $j=1,2,\ldots,M$, is replaced by $\bar{a}_j + p\Delta a_j$, $0 < p \leq 1$. With luck, and a good initial guess, this method will converge to the "best" answer.

This program requires more input information than the linear LEAST SQUARES PROGRAM. In addition to output like the LEAST SQUARES PROGRAM, this program prints information about each iteration.

PIECEWISE QUARTIC FIT

This complete FORAST program approximates a bivariate table of data (x_i,y_i), i=1,2,...,N, with a series of connected quartic polynomials. As many points as possible (without exceeding a specified least squares error) are placed in each succeeding polynomial. The resulting approximating function is continuous and has continuous first derivatives.

CUBIC SPLINE

This complete FORAST program fits N data points (x_i,y_i), i=1,2,...,N, with N - 1 cubic equations. The approximating function passes through each data point, is continuous, and has continuous first and second derivatives.

LEAST SQUARES CUBIC SPLINE

This complete FORAST program fits N data points (x_i,y_i), i=1,2,...,N, with a prescribed number, say K, cubic equations. The user specifies the abscissas for K + 1 "break points". The program finds the K + 1 corresponding ordinates so that the cubic spline function F(x), through these break points, makes $\sum_{i=1}^{N}(y_i - F(x_i))^2$ a minimum.

POLYGONAL CURVE

It is sometimes useful to approximate bivariate data by a series of connected straight lines. This program uses dynamic programming to locate the best values of the independent variable for the ends of a specified number of straight lines. For these "break points", the approximating polygonal curve is the best in the least squares sense.

CONSTRAINED NONLINEAR LEAST SQUARES

Sometimes we want to approximate a set of data (X_i,y_i), i=1,2,...,N, with a function F(X,A) and at the same time satisfy a number, say K, inequality (and/or equality) constraints $G_k(A) \geq 0$ (or $G_k(A) = 0$), k=1,2,...,K. This FORTRAN program finds an approximate answer to this

problem by minimizing the function $U(A) = \sum_{i=1}^{N+K}(f_i(A))^2$, where $f_i(A) = y_i - F(X_i,A)$ for i=1,2,...,N, and $f_{N+k}(A) = G_k(A)$ for unsatisfied constraints ($f_{N+k}(A) = 0$ if $G_k(A)$ is an inequality constraint greater than 0).

The user must supply three subroutines: FCODE to evaluate $f_i(A)$, PCODE to evaluate $\partial f_i/\partial a_j$, j=1,2,...,M, and SUBZ to set up some initial conditions. The program uses a combination of the differential correction method (used by the FORAST NONLINEAR LEAST SQUARES program) and steepest descent to minimize U(A). This method should converge in most cases.

NONLINEAR LEAST SQUARES FOR CORRELATED DATA

This FORTRAN routine is somewhat different than the other least squares routines. The data points consist of $\bar{X}_i = (x_{i1},x_{i2},...,x_{iK})^T$; m_i, the standard error of weight one; and R_j, the correlation matrix of cofactors. The object of the program is to find $\xi_i = X_i - \bar{X}_i$, i=1,2,...,N, and $A = (a_1,a_2,...,a_M)$ so that $U(\xi) = \sum_{i=1}^{N}\xi_i^T(m_i^2R_i)^{-1}\xi_i$ is minimized and $F(X_i + \xi_i,A) = 0$ for i=1,2,...,N. The user must supply a main program which sets up the input and controls the iterations, and a subroutine to compute $F(\bar{X}_i + \xi_i,A)$, and the partial derivatives of this F with respect to ξ_{ik}, k=1,2,...,K, and a_j, j=1,2,...,M.

NLPROG

This is a function minimizing FORTRAN subroutine which permits constraints. If U(A) is the function to be minimized subject to the inequality constraints $G_k(A) \geq 0$, k=1,2,...,J, and the equality constraints $G_k(A) = 0$, k=J+1,J+2,...,J+K, this subroutine is designed to minimize $P(A,\rho) = U(A) + \rho\sum_{k=1}^{J}1/G_k(A) + \sum_{k=J+1}^{J+K}G_k^2(A)/\rho^{\frac{1}{2}}$ over the region $Q = \{A:G_k(A) \geq 0, k=1,2,...,J\}$ for $\rho=\rho_1,\rho_2,...,\rho_L$ with $\rho_r > \rho_{r+1} > 0$

until $\rho_L \sum_{k=1}^{J} 1/G_k(A) < \theta$. If the original estimate for A is not in Q, the routine will first try to find an initial A which is in Q. Actually, NLPROG is one of four different routines depending on the method used to minimize $P(A,\rho)$. Two of these are direct search methods which do not require derivatives. One is a quasi- Newton method which uses first derivatives (the partials of F and G_k with respect to a_j). The other routine uses Newton's method, which requires first and second partial derivatives.

The user must code a main program to supply the initial data and control parameters for NLPROG. He must also supply a subroutine to evaluate F and the G_k and, if needed, a subroutine to evaluate their derivatives.

III. GENERAL DISCUSSION

A. Classification of Routines

In the previous section we characterized the BRLESC function approximation routines as FORAST or FORTRAN routines, as standard subroutines or something else, and as "interpolating", "fitting", or "function minimizing" routines. The separation between FORAST and FORTRAN routines is necessary and definite. The distinction between the standard subroutines and other routines is also useful and quite clear. (The standard subroutines are all on the FORAST compiler tape and equivalent FORTRAN subroutines are available on cards.) The third classification is very arbitrary. The broadest definition of any of the three terms includes all the routines. An "interpolating" routine, by our definition, is one that produces a numerical estimate of the dependent variable for a given set of the independent variables. Thus, we had only one interpolating routine, the standard divided difference interpolation routine. It is convenient to classify the three "function minimizing" routines separately because the user is free to choose the measure of "goodness" of the fit. All the remaining "fitting" routines except the CUBIC SPLINE use the principle of least squares.

There are several other classifications which are convenient for some purposes. We can separate the bivariate routines and the routines that can be used with more than two variables. We can also classify the routines by continuity, smoothness, or by the measure of goodness of the fit. Some of the routines use or permit constraints. These topics will be taken up in the next paragraphs.

B. Goodness of Approximation and Constraints

Approximating functions generally satisfy some set of constraints, optimize some measure of goodness of the fit, or do both. The approximating functions for two of our BRLESC routines, the DVDINT and the CUBIC SPLINE, are completely determined by constraints. For the three function minimizing routines, the function to be minimized is selected by the user (and computed by a user coded program). All the other routines use the principle of least squares and minimize the sum of squares of the residuals, $\sum_{i=1}^{N} R_i^2$.

The user of the function minimizing routines may choose the sum of squares of residuals, $U(A) = \sum_{i=1}^{N} R_i^2$, as the function to minimize. (This is equivalent to minimizing $(\sum_{i=1}^{N} R_i^2)^{\frac{1}{2}}$, the so-called "Euclidean" norm.) Another useful choice is $U(A) = \text{Maximum}|R_i|$, $i=1,2,\ldots,N$, the "Tchebycheff" or "uniform" norm used to find the minimax or Tchebycheff solution. The norm $U(A) = \sum_{i=1}^{N} |R_i|$ is another frequently mentioned choice. The form of the approximating function is selected by the user.

In some of the least squares routines the form of the approximating function is specified by the routine. In others the user may select this function. The residuals for all the least squares routines, except the NONLINEAR LEAST SQUARES FOR CORRELATED DATA program, are the difference between the given value of the dependent variable and the approximated value (i.e., $R_i = y_i - F(X_i,A)$). The measure of goodness of

the fit is the unbiased estimate of the standard deviation, ERMS = $(\sum_{i=1}^{N} R_i^2/(N-M))^{\frac{1}{2}}$, where N is the number of data points and M is the number of coefficients determined by the routine.

The NONLINEAR LEAST SQUARES FOR CORRELATED DATA program is somewhat different. The N data points consist of $X_i = (x_{i1}, x_{i2}, \ldots, x_{iK})^T$, the measured values of the variables; m_i, the standard error of weight one; and R_i, the correlation matrix of cofactors. The coefficients $A = (a_1, a_2, \ldots, a_N)$ and the correction vectors ξ_i, i=1,2,...,N, are found so that $F(X_i+\xi_i, A) = 0$ for all i and $U(\xi) = \sum_{i=1}^{N} \xi_i^T (m_i R_i)^{-1} \xi_i$ is a minimum. $(U(\xi)/(N-M))^{\frac{1}{2}}$ is used as the measure of goodness of the fit.

The FORTRAN routine, CONSTRAINED NONLINEAR LEAST SQUARES permits constraints. For this program the user must compute the constraints G_j, j=1,2,...,J as well as the residuals R_i, i=1,2,...,N and their first derivatives so he may define them as he wishes. The routine minimizes $U^*(A) = \sum_{i=1}^{N} R_i^2 + \sum_{j=1}^{J} G_j^2$. The routine prints $(\sum_{i=1}^{N} R_i^2/(N-M))^{\frac{1}{2}}$ and several other quantities to help determine the validity and goodness of the approximation.

C. Bivariate Routines

Some of the routines are designed to handle many variables but a few are restricted to two variables, a dependent variable y, and an independent variable x. All of these approximate y by a polynomial or set of polynomials in x. The routines restricted to two variables are DVDINT (D.D.IN), POLYLS (P.L.SQ), PIECEWISE QUARTIC FIT, CUBIC SPLINE, LEAST SQUARES CUBIC SPLINE, and POLYGONAL CURVE. The first of these routines is polynomial interpolation. The others are substitutes for polynomial interpolation which may remove or reduce the main objections to interpolation.

1. Interpolation. We have defined an interpolation routine as one which supplies a value of the independent variable. This is a very

narrow definition but it describes the purpose of the DVDINT subroutine quite well From experience we think of interpolation as a process by which we can approximate the value of a dependent variable between values given in a table. If we include approximation for the given points as well, we are describing a process that approximates the tabular function by some other function. In this broader sense, all the routines discussed in this report are interpolation routines.

Any text on numerical methods devotes some space to interpolation. There is usually some mention of interpolation involving more than one independent variable and some discussion about using various functions, including trigonometric functions to interpolate with; but the major portion of the discussion is restricted to interpolation by polynomials. Most of the familiar methods of interpolation (e.g., Aitken's iterative interpolation, Lagrange's interpolation formula, Newton's forward and backward difference methods, divided difference, etc.) are just different ways of evaluating an M-th degree polynomial through M + 1 points near x in a table (x_i,y_i). The Newton's interpolation formula with divided differences was chosen for the standard interpolation routine (DVDINT in FORTRAN, D.D.IN in FORAST). This routine is easy to use, efficient, and accurate. Unfortunately, it is so familiar and well documented that we do not give its use enough thought, let alone check the results. Polynomial interpolation has some drawbacks: If we consider the approximation over the entire range of the table, the function changes at most of the data points and the derivatives fail to exist at these points, even though the function itself is continuous. A high degree polynomial may oscillate wildly and errors in the data tend to augment this oscillation. This is pointed out in the Example. Finally, it is too easy to overlook gross errors in the data.

2. Other Bivariate Routines. The other bivariate routines are, in a real sense, substitutes for polynomial interpolation which remove or reduce one or more of the objectionable features of polynomial interpolation. These routines, except CUBIC SPLINE, use least squares and consequently reduce the effect of small errors. Some of these

routines produce all the residuals, hence, large errors in the data points can be discovered. The other routines use two or more polynomials connected at points we call "break points".

The approximate function from the CUBIC SPLINE routine passes through each data point. A different cubic polynomial is used between each pair of data points. This approximation has continuous first and second derivatives.

The x coordinates of the break points for the LEAST SQUARES CUBIC SPLINE routine must be chosen by the user. The routine chooses the corresponding y coordinates so that the cubic spline function through the break points produces the minimum sum of squares of residuals. This approximating function also has continuous first and second derivatives.

The break points for the POLYGONAL CURVE are selected by the program. (The user must state the number of them to be used.) There are no derivatives at these break points since the straight lines which meet there have different slopes.

The PIECEWISE QUARTIC FIT routine selects its own break points from the data points. As many data points as possible, without causing too large an ERMS error, are included between each break point. The approximating function passes through each of these break points and has a continuous first derivative there.

With the exception of the divided difference interpolation subroutine, the polynomial least squares subroutine, POLYLS, is the easiest of the bivariate routines to use. If a good fit can be made with a low degree polynomial, this is an excellent choice. Unfortunately, even with least squares, approximations with high degree polynomials tend to have undesirable oscillations.

The bivariate routines, with the exception of POLYLS must have the data ordered on the independent variable. The DVDINT subroutine permits either increasing or decreasing ordering. The other routines assume strictly increasing x_i's.

3. Smoothness. Although it is possible to use multivariate approximating functions which have discontinuities or other disagreeable features, the usual function chosen is continuous and has continuous derivatives throughout the entire range of definition. This is not as generally true for bivariate approximating functions. Approximation by polynomial interpolation produces an approximating function which does not have derivatives at most of the data points. A least squares fit over the entire range of the data produces an approximating function with continuous derivatives of all orders; but fitting with a low degree polynomial may give a poor approximation at the data points, and fitting with a high degree polynomial may give good results at the data points but unbelievable intermediate values. The PIECEWISE QUARTIC FIT program produces an approximation that has continuous first derivatives and should not oscillate too wildly. The CUBIC SPLINE and the LEAST SQUARES CUBIC SPLINE programs produce an approximating function that has continuous first and second derivatives. Spline functions have received a great deal of attention recently because of their "smoothness" properties. In particular, the cubic spline function, say $S(x)$, over $N \geq 3$ data points, with $d^2S/dx^2 = 0$ at x_1 and x_N satisfies $\int_{x_1}^{x_N} (d^2S(x)/dx^2)^2 dx \leq \int_{x_1}^{x_N} (d^2f(x)/dx^2)^2 dx$ where $f(x)$ is any continuous function with continuous first and second derivatives which satisfies $f(x_i) = y_i$, $i=1,2,\ldots,N$.

4. Checking. A graph of the approximating function with the original data points superposed gives an excellent qualitative check of approximations in two variables. Gross errors in the input data are evident; and unusual features of the results, such as excessive oscillation or poor approximation in some particular region, are obvious. This check requires extra work, but it is worthwhile.

D. Least Squares

Most of the fitting programs for the BRLESC are based on the principle of least squares applied to a discrete set of points (X_i,y_i), $i=1,2,\ldots,N$. It is assumed there is some function $Y(X)$ for which $Y(X_i) = y_i$. (X_i may be a single variable or a vector, $X_i = (x_{i1},x_{i2},\ldots,x_{iK})^T$.) The function $Y(X)$ is to be approximated by a function $F(X,A)$ where A represents the M unknown parameters $a_1,a_2,\ldots,a_M$. The "best" fit in the least squares sense (for a particular function $F(X,A)$ relative to a non-negative weighting function $w(X)$, over the set of N data points) is achieved when A is found such that $U(A) = \sum_{i=1}^{N} w(X_i)(Y(X_i) - f(X_i,A))^2$ is a minimum. For this to be meaningful, there must be at least as many data points as unknown parameters (i.e., $N \geq M$). We will assume $w(X) = 1$ in what follows. This is the most frequent choice.

1. Linear Least Squares Routines. If $F(X,A)$, the approximating function, is linear with respect to the M unknown parameters $a_1,a_2,\ldots,a_M$ we can write $U(A) = \sum_{i=1}^{N}(y_i - \sum_{j=1}^{J} a_j\varphi_j(X_i))^2$ where the $\varphi_j(X)$, $j=1,2,\ldots,M$, are M suitable functions of X. (As far as the fit is concerned, the $\varphi_j(X)$ do not have to be defined except at the N data points. However, if the result is to be useful they should be well behaved and fairly easy to evaluate for all X in the range of the fit.) The function U(A) will be a minimum if $\partial U/\partial a_k = 0$, $k=1,2,\ldots,M$. This is a system of M linear equations, called "normal equations", in the M unknown a_j:

$$\sum_{j=1}^{M} a_j \sum_{i=1}^{N} \varphi_k(X_i)\varphi_j(X_i) = \sum_{i=1}^{N} \varphi_k(X_i)y_i, \qquad k=1,2,\ldots,M.$$

This can be rewritten in matrix notation as $WA = V$, where A is the vector $(a_1a_2,\ldots,a_M)^T$, W is the symmetric MxM matrix with $\sum_{i=1}^{N} \varphi_j(X_i)\varphi_k(X_i)$

as the element in row j column k, and V is the vector $(v_1, v_2, \ldots, v_M)^T$ with $v_k = \sum_{i=1}^{N} \varphi_k(X_i) y_i$. (If we consider $\varphi_j(X_i)$ to be the j-th element in the i-th row of an NxM matrix P, and y_i the i-th element in a vector Y, we can write $W = P^T P$ and $V = P^T Y$. The normal equations are frequently written in the form $P^T PA = P^T Y$.)

The normal equations can be solved for A ($A = W^{-1} V$) if W is not singular. (The BRLESC routines for solving systems of equations use some form of Gauss elimination.) The accuracy of this solution depends on how well conditioned W is, how large M is, and the number of digits carried by the routine. The BRLESC single precision carries about 16 decimal digits which is equivalent to double precision on most computers.

Classical theory recognizes three situations for linear systems of equations: (1) The equations are inconsistent (e.g., $x + y = 1$, $x + y = 2$), hence no solution exists. (2) The equations are not independent (e.g., $x + y = 1$, $2x + 2y = 2$), in which case an infinity of solutions exist. (3) There is a unique solution. In theory it is easy to tell these cases apart, but in practice machine round-off error blurs the differentiation between the three cases.

A frequent error in using linear least squares is to choose a set of functions, $\varphi_k(X)$, k=1,2,...,M, which are not really independent over the data being fitted. In one's zeal to find an answer quickly, it is easy to overlook even simple dependent relations among functions. The particular set of data and the limitation of machine accuracy sometimes make the choice of clearly independent functions more difficult. For example, if x is greater than 10^5, $x^4 + 1 = x^4$ on BRLESC; hence, x^4 and $x^4 + 1$ are not independent on BRLESC for x greater than 10^5. (Similarly, $x^{-4} + 1 = 1$ for $x > 10^5$.)

The easiest way to get a linear least squares fit is to use the FORAST LEAST SQUARES PROGRAM. This allows a fair amount of flexibility with a minimum of programming and gives "best" coefficients and some results to check the goodness of the fit. Some formula cards and control cards must be prepared, but no coding is needed. The MULTIPLE REGRESSION program is almost as easy to use. It will select a statistically significant submodel, from a candidate model suggested by the user, before carrying out the final fit. The two subroutines GENLSQ and POLYLS (G.L.SQ and P.L.SQ in FORAST) are also easy to use. The user must set up the data in the BRLESC in a specified form, call the subroutine, and then print, or use, the results as he wishes. All these routines produce some results for judging the goodness of the fit. The FNEQS and MATINV (F.N.E. and SY.SNE in FORAST) subroutines require more programming but they permit some additional freedom.

2. Nonlinear Least Squares. If $F(X,A)$, the approximating function, is not linear in the a_j, the minimization of $U(A) = \sum_{i=1}^{N}(y_i - F(X_i,A))^2$ is more difficult. The solution generally requires iteration and sometimes the iteration fails to converge. Three of the programs on our list are called "nonlinear least squares" programs. They are quite different. The simplest of the three programs and the easiest to use is the FORAST NONLINEAR LEAST SQUARES PROGRAM which is similar to the FORAST linear LEAST SQUARES PROGRAM. This program uses an iteration method for minimizing $U(A)$ which is sometimes called the Gauss-Newton method, but is locally called differential corrections.

The FORTRAN routine called CONSTRAINED NONLINEAR LEAST SQUARES is more difficult to use, but it has some features that might make it desirable. It permits constraints by adding the squares of the constraints to the sum of the squares of the residuals. The resulting function is minimized with a combination of the method of steepest descent and the method of differential corrections[19] that should converge in most cases.

The third nonlinear least squares routine is the FORTRAN routine called NONLINEAR LEAST SQUARES FOR CORRELATED DATA. This routine uses N data points in the form: $\bar{X}_i = (x_{i1}, x_{i2}, \ldots, x_{iK})^T$, the measured value of the i-th data point (Notice that no dependent variable is prescribed.); m_i, the standard error of weight one; and R_i, a KxK correlation matrix of cofactors. The nonlinear relation F(X,A) selected by the user is assumed to be exact at each of the N points. The only errors being errors in each of the NK variables x_{ik} (i=1,2,...,N; k=1,2,...,K).

E. Function Minimizing Routines

If the approximating function F(X,A) is "best" when some function U(A) is a minimum, the function minimizing routines are a rather natural choice of routine for finding the "best" value of A. In the case of least squares, $U(A) = \sum_{i=1}^{N} (R_i)^2$, where R_i is the i-th residual. This choice of U(A) is frequently chosen because it is meaningful and easy to analyze. More efficient routines are available for linear least squares problems, but the subroutines FNMIN and FDMIN have been used for some nonlinear least squares problems. Two other fairly common choices of U(A) for discrete data are $U(A) = \sum_{i=1}^{N} |R_i|$ and U(A) = Maximum $|R_i|$. Other useful choices are possible. FNMIN is the easiest of the function minimizing routines to use. The user must code a subroutine that supplies U(A) for any given A. FDMIN requires $\partial U(A)/\partial a_j$, j=1,2,...,M, in addition to U(A). Neither of these standard subroutines permit constraints. One might reach a satisfactory solution for a constrained problem by adding a penalty function to U(A), but the nonstandard FORTRAN subroutine NLPROG is probably a better choice. As an illustration of a penalty function, suppose we want to minimize $U^*(A) = \sum_{i=1}^{N} R_i^2$ but have constraints $G_r(A) = 0$, r=1,2,...,S. We can define $U(A) = \sum_{i=1}^{N} R_i + \sum_{r=1}^{S} w_r G_r^2(A)$ where the w_r are positive weights.

By adjusting the weights w_r, it is possible to obtain an acceptable approximation. The CONSTRAINED NONLINEAR LEAST SQUARES routine uses such a method. The NLPROG routine uses this method to handle equality constraints and a sum of the form $\sum w_r/G_r(A)$ for inequality constraints.

F. Example

This simple example was chosen to illustrate the effect of errors in data upon interpolation. The table of data (1,1), (2,4.4), (2.1,4), (3,9) was generated by perturbing the y values of $y = x^2$. The data could realistically be for this function if the readings were accurate to within .5 or if they were accurate to within .01 with the y values for x = 2.0 and x = 2.1 reversed. What is unrealistic is that the "true" function is known.

The approximating functions corresponding to linear, quadratic, and cubic interpolation, the cubic spline, and the best least squares quadratic were found. The DVDINT does not specifically display its approximating function as we do here.

Linear Interpolation

$F(x) = 3.4x-2.4 \qquad 0 \le x \le 2$

$F(x) = -4x+12.4 \qquad 2 \le x \le 2.1$

$F(x) = (50x-69)/9 \qquad 2.1 \le x \le 3.55$

Quadratic Interpolation

$F(x) = (-740x^2+2594x-17440)/110 \qquad 0 \le x < 2.1$

$F(x) = (860x^2-3886x+4728)/90 \qquad 2.1 \le x \le 3.55$

Cubic Interpolation

$F(x) = (8060x^3-47766x^2+90244x-49548)/990 \qquad 0 \le x \le 3.55$

Least Squares Quadratic

$$F(x) = .99992444x^2 - .0199849x + .03834471$$

Cubic Spline Fit

$$F(x) = -3.588964818(x-1)^3 + 6.98896418x - 5.988964818 \qquad 1 \le x \le 2$$

$$F(x) = 35.88964818(x-2.1)^3 + 49.57226003(x-2)^3 - 4.854619082x + 14.14512781 \qquad 2 \le x \le 2.1$$

$$F(x) = 5.50802889(3-x)^3 + 10.01705895x - 21.05117688 \qquad 2.1 \le x \le 3$$

The following list shows $F(x)$ and $R(x) = F(x) - x^2$ at $x = 0$, 1.5, and 2.5 for each of these approximating functions.

Method	F(0) = R(0)	F(1.5)	R(1.5)	F(2.5)	R(2.5)
Linear Int.	- 2.4	2.7	.45	6.222	- .028
Quad. Int.	-15.8	4.382	2.132	4.311	-1.939
Cubic Int.	-50.0	5.603	3.353	3.497	-2.753
L. Sq. Quad.	.038	2.258	.008	6.238	- .012
Cubic Spline	- 2.4	4.046	1.796	4.680	-1.570

This table shows that increasing the degree of the interpolating polynomial will not necessarily produce better estimates of the "true" function. The extrapolation to $x = 0$ demonstrates why extrapolation from tabular data is so universally condemned.

From the viewpoint of approximating x^2 from the table of data, the least squares quadratic was fairly successful. However, if we consider this fit without a priori knowledge of the "true" function, it too is quite poor. The estimated standard error, ERMS, is .572. This is a rather large error for numbers between 1 and 9. The following table of coefficients "sigmas and t's" shows that the fit is really not very significant. a_2 and a_3 have no significance and even a_1 is of doubtful validity. (A frequently used rule-of-thumb is to discard all coefficients with corresponding t's less than 2.) The residuals for the four points are -.02, .40, -.40, and .02, respectively.

j	a_j	σ_j	t_j
1	.9999	.575	1.738
2	-.0200	2.164	-.009
3	.0383	2.322	.018

With only four points we can tell very little about the data from the residuals, but suppose we investigated the data and discovered that the y_2 and y_3 had been interchanged. The best least squares quadratic fit for the four points (1,1), (2,4), (2.1,4.4), (3,9) is $F(x) = 1.00496184x^2-.02009923x+.01536407$.

This produces residuals (.00023, -.00499, .00504, -.00028), ERMS = .0071, and

j	a_j	σ_j	t_j
1	1.004962	.00714	140.8
2	-.020099	.02881	-.698
3	.01536407	.02685	.572

Again a_2 and a_3 are not significant, but a_1 is. If we refit without a_2 and a_3 we get $F(x)=.9996245x^2$ with residuals (-.00038, -.0015, .0083, -.0034) and ERMS = .0053. Notice that the residuals are generally larger, but the unbiased estimate of error has dropped from .0071 to .0053 with the deletion of a_2 and a_3.

IV. DESCRIPTION OF ROUTINES

The use of each of the routines will be discussed in this section. The details of implementing the routines are presumably contained in the program write ups, most of which are included in the Appendix. What I hope to do is briefly describe what the routine is supposed to do, indicate the amount of work necessary to use the routines, point out any unusual difficulties, and add any other remarks that seem pertinent.

DVDINT[5,6] (D.D.IN[3] in FORAST)

This standard subroutine uses Newton's divided difference method of polynomial interpolation. The data (x_i,y_i), i=1,2,...,N, must be ordered on the dependent variable x, either strictly monotone increasing or strictly monotone decreasing. The value of x for which F(x), the approximating function, is desired must be between $x_1-(x_2-x_1)$ and $x_N+(x_N-x_{N-1})$. The approximating function is an (M-1)th degree polynomial through M points near x. The resulting function is continuous and passes through each of the data points. The derivative however is not continuous. It fails to exist at most of the interior points.

The most frequent error made with this subroutine is attempting to extrapolate outside the acceptable range. The program produces an error and halts when this occurs. Occasionally someone specifies M greater than N and gets the same error print. It is my opinion that if M greater than three is used, the user should not only substantiate his choice, but also produce a graph of the fitted function with the data superposed. The inspection of such a graph can also reveal errors in the input data.

Instructions for using this subroutine are in the appendix.

MATINV, FNEQS[5,6] (S.N.E., F.N.E. etc.[3] in FORAST)

The standard subroutine for solving a system of linear equations, MATINV, can be used to solve the "normal equations" associated with linear least squares. For linear least squares the approximating function is linear in the M unknown coefficients. That is $F(X,A) = \sum_{j=1}^{M} a_j\varphi_j(X)$ where the $\varphi_j(X)$ are appropriately chosen, usually well behaved functions. We wish to find $A = (a_1,a_2,\ldots,a_M)$ that minimizes $U(A) = \sum_{i=1}^{N}(y_i - \sum_{j=1}^{M} a_j\varphi_j(X_i))^2$. Such an A satisfies

$\partial U(A)/\partial a_k = 0$, $k=1,2,\ldots,M$, or

$\sum_{j=1}^{M} a_j \sum_{i=1}^{N} \varphi_k(X_i)\varphi_j(X_i) = \sum_{i=1}^{N} \varphi_k(X_i)y_i$, $k=1,2,\ldots,M$. These are the so-called "normal equations". The formation of the matrix of coefficients for these normal equations is usually best accomplished by using FNEQS, the form normal equations subroutine. These programs do not contain checking features, except for an error print from MATINV if a singular matrix occurs. The user is advised to use GENLSQ or POLYLS subroutines or the LEAST SQUARES PROGRAM instead.

One frequently made error in using linear least squares fitting procedures is to choose functional terms which are not really independent over the given set of data. The result can be an equation that fits the particular set of data used, but is useless for intermediate points. If possible, it is useful to reserve some data for an independent check of the fit. A reasonable rule-of-thumb is to have three or four data points for each unknown coefficient. If M is large, the accumulated roundoff error in inverting even well behaved matrices may produce inaccurate results.

Instructions for using these subroutines are in the Appendix.

GENLSQ and POLYLS (G.L.SQ and P.L.SQ in FORAST)

The GENLSQ, general least squares, routine is a standard subroutine for making a linear least squares fit. The input requirements are the matrix $(\varphi_j(X_i))$, $(j=1,2,\ldots,M;\ i=1,2,\ldots,N)$ and the vector of the dependent variable, $(y_1,y_2,\ldots,y_N)^T$, where the desired approximating function is $F(X,A) = \sum_{j=1}^{M} a_j\varphi_j(X)$. The POLYLS, polynomial least squares, is a modification of GENLSQ for polynomials in one dependent variable. In this case, the user need only supply the two vectors $(x_1,x_2,\ldots,x_N)^T$ and $(y_1,y_2,\ldots,y_N)^T$, and the subroutine automatically uses $\varphi_j(x_i) = x_i^{j-1}$. (The FORAST version allows more flexibility in choosing terms and permits weighting and omitting data points.)

These routines will compute a_j, $j=1,2,\ldots,M$, which are "best" in the least squares sense. That is, $U(A) = \sum_{i=1}^{N}(y_i - F(X_i,A))^2$ is a minimum. These routines will compute the approximate values of the function $F(X_i,A)$, the residuals $R_i = y_i - F(X_i,A)$, the unbiased root-mean-square error $ERMS = \left(\sum_{i=1}^{N}(R_i^2)/(N-M)\right)^{\frac{1}{2}}$, the estimates of error for each coefficient σ_j, and an estimate of the significance of the coefficients $t_j = a_j/\sigma_j$.

These routines are easy to use although it is easy to omit or misplace one of the many dummy variables. However, as with most programs, the most troublesome and very common errors are errors in the data. The problems with functional dependence and errors in matrix inversion if M is large are still present. An independent check of the resulting approximation is always desirable.

Instructions for using these subroutines are in the Appendix.

FNMIN and FDMIN[5,8] (FN.MIN[9] and FD.MIN[10] in FORAST)

These two standard subroutines are function minimizing routines. They can be used to attempt to solve function approximation problems. Suppose we wish to find $A = (a_1,a_2,\ldots,a_M)^T$ which makes U(A) a minimum. These two subroutines attempt to minimize U(A) directly. FNMIN uses a direct search technique which does not require derivatives. FDMIN uses a quasi-Newton method which does require first derivatives of U(A) with respect to each of the a_j.

The user must supply the initial guess for A, some control parameters, and a subroutine to evaluate U(A), (U(A) and $\partial U(A)/\partial a_j$, $j=1,2,\ldots,M$, for FDMIN). The programs are easy to use. However, they are rather slow (compared to linear least squares) and convergence to the "best" answer is uncertain. If the location of the minimum of U(A) is critical, it is desirable to check the results of these minimizing

programs. If the function U(A) is sufficiently well behaved, some analysis may confirm the results, but this is not always possible. One partial check is to repeat the minimization from several different initial positions. If the same solution is reached several times, the existence of at least a local minimum there is fairly well established.

Most of the difficulties with these two routines have been from errors in programming the evaluation of the function or its derivatives, or from using a U(A) which does not have a minimum for finite A.

Neither of these routines accept constraints. Constraints can sometimes be included by using a penalty function, but the more general, nonstandard, FORTRAN subroutine NLPROG should be a better choice.

Instructions for using these subroutines are in the Appendix.

LEAST SQUARES PROGRAM[12], MULTIPLE REGRESSION[13], NONLINEAR LEAST SQUARES[15*]

These three routines are grouped together because of their many similarities. All three are complete FORAST programs (programmed by L. W. Campbell) which need the same type of formula definition cards and control cards. All of them have been modified since their respective descriptions were written. In particular, the restrictions on the number of unknowns has been greatly increased in every case. It is desirable to get a program deck directly from System Programming, Computer Service Division, along with information as to just what options and changes it contains. It might also be wise to ask someone from Systems Programming to check the completeness and correctness of the first set of cards prepared for one of these programs.

Despite some uncertainty as to just what the programs will do and their exact implementation, these programs should be given first consideration for any least squares fitting project. The fact that they are complete programs is both an advantage and a disadvantage. Since

*FORTRAN programs for MULTIPLE REGRESSION and NONLINEAR LEAST SQUARES are available.

they are complete programs, no coding is required (formula cards and control cards are used instead) and hence, no knowledge of FORAST (or FORTRAN) is needed. Also, the results usually receive at least a cursory check before being used for further computation. On the other hand, the results of these programs are tabulations (or at best numbers on cards) and are not directly available for use in other routines, as the output of a subroutine would be.

In all three programs, the parameters which control the fitting are punched on cards. Each data point, say (V1,V2,...,Vk) is read in under the control of a FORMAT which may have to be changed. The user defines the form of the equation by "formula control" cards. These formula control cards may contain arithmetic operations and some subroutines: SIN, COS, SQRT, LOG, EXP, ARCTAN and ARCSIN. For example, V1*EXP(V6) = V2**3+LOG(V3)+(V4+V5+SQRT(V7))$. This corresponds to the formula $V1e^{V6}="a_0"+a_1V2^3+a_2\ell n_e(V3)+a_3(V4+V5+V7^{\frac{1}{2}})$. (The constant term, a_0, would be automatically included in the multiple regression program, unless specifically omitted, but not in the other two programs.) The output for all of these programs includes the coefficients, approximate function values, residuals, ERMS, and the "sigmas and t's" which indicate the variance and significance of the coefficients.

The LEAST SQUARES PROGRAM is similar to the GENLSQ subroutine. There is an option which simplifies the input if a polynomial fit with one dependent variable is wanted. The LEAST SQUARES PROGRAM is still available but the Computer Support Division now recommends using the MULTIPLE REGRESSION program instead.

The MULTIPLE REGRESSION program treats the model prescribed on the formula control cards as a candidate model. The program selects a statistically significant submodel and gives coefficients, residuals, etc., for this choice. A list of the terms as they are added or removed from the regression model and the change in ERMS is also given. Modifications for graphing and for an input tape for a companion program, "Prediction Intervals for Estimates from Linear Regression Models",[14] are described in a description dated Nov. 1967.

This very useful program is thoroughly described in Reference 13.

The NONLINEAR LEAST SQUARES program is similar to the LEAST SQUARES PROGRAM. Since the formula is not linear in the unknown coefficients, an iterative scheme, locally called "differential corrections", is used to attempt to minimize $U(A) = \sum_{i=1}^{N}(y_i - F(X_i,A))^2$. $F(X_i,A)$ is expanded in Taylor's series about the latest estimate of A, say $\bar{A}$, discarding quadratic and higher terms. $\Delta A = (\Delta a_1, \Delta a_2, \ldots, \Delta a_M)$ is found so that $U^*(\Delta A) = \sum_{i=1}^{N}(y_i - F(X_i,\bar{A}) - \sum_{j=1}^{M}\Delta a_j \partial F(X_i,\bar{A})/\partial a_j)^2$ is a minimum, as in linear least squares programs. Then, $\bar{A}$ is replaced by $\bar{A} + p\Delta A$, $(0 \leq p \leq 1)$ and the process is repeated until each $|\Delta a_j| < \epsilon_j$, or the prescribed maximum number of iterations is exceeded.

The input for this routine is more involved than that for the last two programs. $F(X,A)$, the derivatives $\partial F(X,A)/\partial a_j$, $j=1,2,\ldots,M$, and the form $y_i - F = \sum_{i=1}^{M}\partial F/\partial a_j$ must all be described. The user must also supply $(\epsilon_1, \epsilon_2, \ldots, \epsilon_M)$ and initial values of A. The output of the program includes fairly detailed results for each iteration as well as $F(X_i,A)$, the residuals $y_i - F(X_i,A)$, and the "sigma and t" values for the final iteration.

This program is also fairly easy to use. The fact that it is not a subroutine is probably a good thing. The resulting approximation should be examined critically before it is accepted. If one must have a subroutine, he can code his own program using the differential correction method and available linear least squares subroutines (or he can try one of the function minimizing subroutines).

Instructions for using NONLINEAR LEAST SQUARES are in the Appendix. These instructions refer to Reference 13.

The author has used the MULTIPLE REGRESSION program, but has no direct experience with the other two programs in this section. Several mistakes were made through carelessness: A FORMAT was wrong, a comma was omitted after a redefinition formula, and a plus sign was lost when a formula card was duplicated. Despite these minor troubles, which point out the necessity for checking all computer input no matter how simple, the MULTIPLE REGRESSION program was relatively easy to use.

PIECEWISE QUARTIC FIT, CUBIC SPLINE[17], LEAST SQUARES CUBIC SPLINE

These three routines have many common features: They are all complete FORAST programs. They are bivariate routines which serve as alternatives to interpolation. The approximating function, a piecewise low order polynomial, has continuous first derivatives (the spline functions also have continuous second derivatives). Two conditions must be stipulated for each routine in addition to the table of data points (x_i,y_i), i=1,2,...,N. The data must be arranged so that the independent variable, x, is strictly increasing (this may not be strictly necessary for the LEAST SQUARES CUBIC SPLINE).

The PIECEWISE QUARTIC FIT uses a set of quartic equations to approximate the table of data. The derivative dy/dx must be prescribed at the first and last point. The approximating function passes through the first and last point and through each "break point", the point where the approximating function changes from one polynomial to another. The derivative at each break point is set equal to the slope of the least squares quadratic polynomial involving the break point and two points on each side of it. The break points are chosen so that as many data points as possible are included in the least squares fit for each succeeding polynomial without exceeding a prescribed root-mean-square error. Finally, if the coefficients of the quartic equation are not significant enough, the quartic polynomial is replaced by a cubic polynomial. This program should produce a very useful fit. I have had no experience with it.

The CUBIC SPLINE produces an approximating function which passes through each data point, is continuous, and has continuous first and second derivatives. Because of this continuity, this approximation function may be preferable to polynomial interpolation. However it still has the defect, in common with polynomial interpolation, that errors in the data may be magnified in the approximating function. The approximating function is a different cubic equation between each pair of data points. One obvious disadvantage of this routine is the creation of a lot of data. The table of data containing 2N numbers is replaced by a table of 5(N-1) numbers: the independent variable in the original data, and coefficients for N-1 cubic equations.

The LEAST SQUARES CUBIC SPLINE routine alleviates the two main disadvantages of the CUBIC SPLINE routine by replacing the constraint that the approximating function must pass through each data point by a least square condition. The user must decide how many cubic equations he wishes, say K of them, and define the abscissas of the break points, say $x^*_1, x^*_2, \ldots, x^*_{K+1}$. It may be difficult to choose good break points. Common sense dictates that a fair number of data points be between each break point, that $x^*_1 \leq x_1$ and $x^*_{K+1} \geq x_N$, and that more break points are needed where the slope of the data is changing rapidly. A cubic spline approximation, say $F(x,A)$, is made through the points (x^*_k, y^*_k), $k=1,2,\ldots,K+1$. The points y^*_k are chosen by the routine so that $\sum_{i=1}^{N}(y_i - F(x_i,A))^2$ is a minimum.

The choice of good break points might prove to be troublesome, but this appears to me to be the most attractive of the three routines described in this section. It should be used more often.

At the present time, the only source of these routines is the Firing Tables Branch of EBL. All three programs are currently being updated. Updated programs and descriptions of them may be available by the time this report is published.

POLYGONAL CURVE[16]

This is a very specialized type of curve fitting. The bivariate table of data (x_i,y_i), i=1,2,...,N is replaced by a function, F(x,A), which is a specified number, say K, of connected straight lines. The POLYGONAL CURVE program chooses break points in a very elegant manner by combining dynamic programming and least squares to produce a polygonal curve which has the "best choice" of break points (i.e., corners) so that $\sum_{i=1}^{N}(y_i-F(x_i,A))^2$ is a minimum. Reference 16 describes the theory for these programs and gives some examples of results. Several FORAST programs have been coded.

If a program deck or additional information is desired, contact Mr. C. M. Frank, Army Materiel Systems Analysis Agency.

CONSTRAINED NONLINEAR LEAST SQUARES[19]

This is the only routine in our list which was not developed at BRL. This routine is a modification of a FORTRAN Share program developed by D. W. Marquardt. It was obtained from Dr. Marquardt by Hue McCoy, Firing Tables Branch, EBL. A few simple changes to make the program compatible with the BRLESC FORTRAN were made by the author of this report. At this time, the only problem which has been run locally, with this program is a simple test problem. This program has been included in our list because it has two features not available in our other least squares programs: It permits constraints, and it uses a method of solution which should be efficient and converge to the "best" answer.

The constraints are included in the problem by simply adding the squares of the unsatisfied constraints to the usual least squares measure of goodness of fit. If there are K constraints, $G_k(A)$, the program minimizes $U(A) = \sum_{i=1}^{N+K} f_i^2$, where $f_i = y_i-F(X_i,A)$ for i=1,2,...,N, and $f_{N+k} = G_k(A)$ or $f_{N+k} = 0$ if $G_k(A)$ is a satisfied inequality constraint.

The method of minimizing U(A) is a combination of the "differential correction" method, used with the FORAST NONLINEAR LEAST SQUARES program, and the well known method of steepest descent. (See Reference 19.)

The user of this FORTRAN program must code three subroutines: FCODE to evaluate f_i, PCODE to evaluate $\partial f_i/\partial a_j$, j=1,2,...,M, and SUBZ to compute parameters needed by FCODE and PCODE.

This program has several attractive features. The two principal ones being the ability to include constraints and the higher probability of convergence than the FORAST NONLINEAR LEAST SQUARES program. The program also supplies statistical results with which to analyze the approximation. The user's subprogram computes each f_i and $\partial f_i/\partial a_j$. This gives great flexibility to the definition of the approximating function; hence, this program can be very versatile. For example, it should be possible to use the program to attempt to solve a system of nonlinear equations.

The program has two principal disadvantages: It requires a considerable amount of coding and input, and anyone using the program is on their own since no one locally has had experience with it.

At present, the only source of a program deck or description is Hue McCoy of EBL or the author of this report.

NONLINEAR LEAST SQUARES FOR CORRELATED DATA[18]

This program is unique among the available BRLESC least squares routines in that it assumes correlation of errors in the data. (A basic assumption for all the other routines is that such correlation does not exist.) This program assumes that the approximating function F(X,A) is correct as to form, and $F(X_i,A) = 0$ if $X_i = (x_{i1},x_{i2},\ldots,x_{iK})^T$ and $A = (a_1,a_2,\ldots,a_M)$ are correct.

The input for each data point consists of $\bar{X}_i = (\bar{x}_{i1},\bar{x}_{i2},\ldots,\bar{x}_{iK})^T$, the i-th estimate of the variables; m_i, the standard error of weight one; and R_i, the KxK correlation matrix of cofactors. The routine

attempts to find $\xi_i = (\xi_{i1},\xi_{i2},\ldots,\xi_{iK})^T$, i=1,2,...,N and $A = (a_1,a_2,\ldots,a_M)^T$ so that $F(X_i + \xi_i,A) = 0$, i=1,2,...,N, and $U(\xi) = \sum_{i=1}^{N} \xi_i^T (m_i^2 R_i)^{-1} \xi_i$ is a minimum.

The user must code a MAIN program which supplies the data and controls the iteration of a subroutine COLSA (iteration is necessary if F(X,A) is nonlinear) and the calling of a subroutine COLSB when a satisfactory fit is found. The user must also code a subroutine to supply $F(X_i + \xi_i,A)$ and the first partial derivatives of this F with respect to ξ_{ik}, k=1,2,...,K, and with respect to a_j, j=1,2,...,M.

It is hard to draw any conclusions about a program without using it, but the following things are obvious: A considerable amount of coding must be done. The program is restricted to five independent variables $(x_{i1},x_{i2},\ldots,x_{i5})$ and even with this restriction, 31 numbers are needed for each data point. On the other hand, the output of COLSA and COLSB are extensive and fairly self explanatory. Finally, this is the only BRLESC routine available to perform least squares fits with correlated data.

NLPROG[11]

This FORTRAN subroutine (actually a group of subroutines) was designed to solve the general nonlinear programming problem: Minimize U(A) subject to J inequality constraints $G_k(A) \geq 0$, k=1,2,...,J and the K equality constraints $G_k(A) = 0$, k=J+1,J+2,...,J+K. The $G_k(A)$, k=1,2,...,J+K, and U(A) can be nonlinear functions of $A = (a_1,a_2,\ldots,a_M)$. This problem is replaced by the set of subproblems: Minimize $P(A,\rho) = U(A) + \rho \sum_{k=1}^{J} 1/G_k(A) + \sum_{k=J+1}^{J+K} G_k^2(A)/\rho^{\frac{1}{2}}$ over A such that $G_k(A) \geq 0$, k=1,2,...,J, for $\rho = \rho_1 \geq \ldots \geq \rho_L$ until $\rho_L \sum_{k=1}^{J} 1/G_k(A)$ is small enough. There are four versions of this subroutine depending on the method used

to minimize $P(A,\rho)$. The user codes a main program to supply data, initial guesses at A, and a number of other parameters. He will also have to code a subroutine to evaluate U(A) and the constraint equations. Depending on which NLPROG version is used, the user may have to supply first and second derivatives of U and each of the G_k, just first derivatives, or no derivatives at all.

This subroutine is nonstandard, it may be difficult to understand, it requires a fair amount of programming and it will be difficult to check the validity of the solution. On the other hand, it can be applied to a wide variety of problems. This subroutine may converge slightly better than the standard minimizing subroutines, FNMIN and FDMIN, but these two subroutines are much easier to use.

At present, the only source of program decks is the author. A complete description of this routine is included in Reference 11. Part of that description is duplicated in the Appendix of this report.

REFERENCES

1. Lloyd W. Campbell and Glenn Beck, "The Instruction Code for the BRL Electronic Scientific Computer (BRLESC)," Ballistic Research Laboratories Memorandum Report No. 1379, November 1961.

2. Lloyd W. Campbell and Glenn Beck, "BRLESC I/II FORTRAN," Aberdeen Research and Development Center Technical Report No. 5, March 1970.

3. Lloyd W. Campbell and Glenn Beck, "The FORAST Programming Language for ORDVAC and BRLESC (Revised)," Ballistic Research Laboratories Report No. 1273, March 1965. [Modification for BRLESC II, 9 October 1968*].

4. F. B. Hildebrand, "Introduction to Numerical Analysis," (McGraw-Hill Book Company, New York, 1956).

5. Listing of FORTRAN Subprogram Card Decks Available from Systems Programming, Bldg. 328, Room 213, Aberdeen Proving Ground, Maryland, 25 July 1968*.

6. Lloyd W. Campbell, "BRLESC FORTRAN Changes," 30 June 1967*.

7. Lila Butler, "FORTRAN General and Polynomial Least Squares Subroutines," 17 July 1968*.

8. Donald F. Taylor, "Function Minimizing FORTRAN Subroutines (On Cards)," 27 March 1968*.

9. Donald Taylor and John Wortman, "A Function Minimizing Subroutine Without Calculating Derivatives," January 1966*.

10. J. C. Torrey and John Wortman, "A Function Minimizing Routine," November 1965*.

11. John Wortman, "NLPROG (A Set of FORTRAN Programs to Find the Minimum of a Constrained Function," Ballistic Research Laboratories Memorandum Report No. 1958, January 1969.

12. Lloyd Campbell, "Least Squares Program with Formula Control Cards," June 1965*. (Amendment, August 1965*.)

13. Harold J. Breaux, Lloyd W. Campbell, and John C. Torrey, "Stepwise Multiple Regression, Statistical Theory and Computer Program Description," Ballistic Research Laboratories Report No. 1330, July 1966. (See modification of Nov. 1967* also Reference 14).

REFERENCES (Continued)

14. Harold Breaux, Kenneth Breitbart, and Lloyd Campbell, "Prediction Intervals for Estimates from Linear Regression Models," February 1968*.

15. Lloyd Campbell, "Nonlinear Least Squares Program," December 1966*.

16. Clinton M. Frank and Ralph E. Shear, "Function Approximation by a Polygonal Curve," Ballistic Research Laboratories Report No. 1363, April 1967.

17. Palmer R. Schlegel, "The Cubic Spline - A Curve Fitting Procedure," Ballistic Research Laboratories Report No. 1253, July 1964.

18. Aivars Celmins, "Light Refraction by a Blast Bubble, (Appendix)," Ballistic Research Laboratories Report No. 1360, March 1967.

19. D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," J. Soc. Indust. and Appl. Math., II, No. 2, (1963) 431-441.

*Unpublished report distributed in BRL. Copies are available from Systems Programming, room 213 building 328. May be included in Reference 2.

TABLE OF CONTENTS FOR APPENDIX

APPENDIX: INSTRUCTIONS FOR USING APPROXIMATION ROUTINES

INTRODUCTION

These descriptions are copied from various published and unpublished sources. I wish to thank L. W. Campbell, CSD, for permission to use material from his publications and other publications from Systems Programming. I also thank Dr. A. Celmins, AMD, for the description of his NONLINEAR LEAST SQUARES ROUTINE FOR CORRELATED DATA, and Joe Hurff, EBL, for the description of LEAST SQUARES CUBIC SPLINE.

Standard FORTRAN Subroutines

These descriptions are taken from an unpublished Systems Programming listing July 25, 1968. Reference 11 of this report has a more detailed description of GENLSQ and POLYLS.

LISTING OF FORTRAN SUBPROGRAM CARD DECKS AVAILABLE FROM
SYSTEMS PROGRAMMING,BLDG,328, RM 213, APG,MD.

KEY TO STAT. NO. FIELD,

- (I) INDICATES INPUT ARG., CALLING PROG. SUPPLIES VALUE
- (R) INDICATES RESULT. SUBPROGRAM STORES VALUE THERE.
- (T) INDICATES TEMPORARY STORAGE.
- (IR) INDICATES ARGUMENT USED AS INPUT AND RESULT.
- (F) INDICATES ARG. USED AS A FUNCTION NAME.
- (S) INDICATES ARG. USED AS A SUBROUTINE NAME.
- (U) INDICATES ARG. WITH UNUSUAL USAGE.

IMPLIED TYPE OF DUMMY ARGUMENT INDICATES REQUIRED TYPE OF ACTUAL ARGUMENT, EXCEPT WHERE NOTED OTHERWISE.

IMPORTANT BRLESC1 RESTRICTIONS-NO. OF DIMENSIONS MUST BE THE SAME BETWEEN ACTUAL AND DUMMY ARGUMENTS. DUMMY ARRAY ARG. CANNOT HAVE ACTUAL ARG. THAT HAS SUBSCRIPT. (ACTUAL ARG. MUST BE JUST ARRAY NAME WHEN DUM. ARG. IS ARRAY.)

```
      SUBROUTINE DVDINT(X,FX,XT,FT,NP,ND)
C           DOES DIVIDED DIFFERENCE INTERPOLATION.
C (I) X   IS ARGUMENT FOR WHICH FUNCTIONAL VALUE IS DESIRED.
C (R) FX  IS NAME OF THE RESULT.
C (I) XT  IS ARRAY OF X VALUES.(1 DIMENSIONAL)
C (I) FT  IS ARRAY OF FUNCTIONAL VALUES.(1 DIMENSIONAL)
C (I) NP  IS THE NUMBER OF VALUES IN XT AND FT ARRAYS.
C (I) ND  IS THE NUMBER OF POINTS TO USE FOR EACH INTERPOLATION.
```

```
      SUBROUTINE MATINV(A,N,C,NMAX,K,DET)
C             MATRIX INVERSION. A=A**(-1)
C(IR)    A IS THE MATRIX AND IS REPLACED BY ITS INVERSE.
C (I)    N IS THE DIMENSION OF THE MATRIX.
C (R)    C IS USED ONLY WHEN K=1 AS DESCRIBED BELOW.
C (I)    NMAX IS THE MAX. NO. OF ROWS OF A AS DECLARED.
C (I)    K DESCRIBES OPTIONS. K=0 MEANS AN N X N MATRIX.K=1 MEANS
C          AN N X N MATRIX AND A SINGLE VECTOR AT C.(THE SOLUTION
C          VECTOR REPLACES THE C VECTOR). K>=2 MEANS AN N X (N+K-1)
C          MATRIX. (THE (K-1) VECTORS ARE REPLACED BY THE (K-1)
C          SOLUTION VECTORS).
C (R)    DET IS THE VALUE OF THE MATRIX DETERMINANT.
C          WHEN A IS SINGULAR, AN ERROR PRINT AND RETURN WITH THE
C          VALUE OF DET SET TO ZERO IS EXECUTED.

      SUBROUTINE FNEQS(A,N,C,NMAX,W)
C        FORM NORMAL EQUATIONS  (FULL N X (N+1) MATRIX).
C (R)    A IS THE MATRIX OF NORMAL EQUATIONS BEING FORMED.
C             A MUST BE CLEARED TO ZEROS BEFORE FIRST CALL OF FNEQS.
C (I)    N IS THE NO. OF TERMS(EXCLUDING FUNCTIONAL VALUE).
C (I)    C IS A VECTOR CONTAINING THE TERMS OF THE EQUATION INCLUDING
C             THE FUNCTIONAL VALUE AS THE LAST TERM.
C (I)    NMAX IS THE MAX. NO. OF ROWS OF A AS DECLARED.
C (I)    W IS THE WEIGHT TO BE APPLIED TO THIS EQUATION.

      SUBROUTINE GENLSQ(X,NRX,F,M,A,NRA,N,C,R,AF,ERMS,SIG,T,DET,IC)
C           USES FNEQS AND MATINV SUBROUTINES.(MUST INCLUDE CARDS.)
C (I)    X    IS A MATRIX OF TERMS OF EQUATIONS.
C (I)    NRX  IS NUMBER OF ROWS IN X.
C (I)    F    IS A VECTOR OF FUNCTION VALUES FOR EQUATIONS.
C (I)    M    IS NUMBER OF EQUATIONS.
C (T)    A    IS A MATRIX OF AT LEAST (N)X(N+1),IS REPLACED BY INVERSE.
C (I)    NRA  IS NUMBER OF ROWS IN A.
C (I)    N    IS NUMBER OF TERMS NOT INCLUDING FUNCTION VALUE, N.LE.99.
C (R)    C    IS A VECTOR FOR N COEFFICIENTS.
C (R)    R    IS A VECTOR FOR M RESIDUALS.
C (R)    AF   IS A VECTOR FOR M APPROXIMATE FUNCTIONS.
C (R)    ERMS IS THE ROOT MEAN SQUARE ERROR,EQUALS ZERO IF M.LE.N.
C (R)    SIG  IS A VECTOR FOR N SIGMAS.
C              SIG IS INVERSE ELEMENT IF INV. ELEMENT IS NEGATIVE.
C (R)    T    IS A VECTOR FOR N T VALUES.
C (R)    DET  IS THE VALUE OF THE DETERMINANT.
C (I)    IC   IS THE CONTROL --
C          IC IS 0   COMPUTE EVERYTHING.
C          IC IS 1   COMPUTE ONLY COEFFICIENTS.
C          IC IS 2   COMPUTE EVERYTHING EXCEPT RESIDUALS AND APPROXIMATIONS.
C          IC IS 3   COMPUTE EVERYTHING EXCEPT APPROXIMATIONS.
```

```
      SUBROUTINE POLYLS(X,F,M,A,NRA,N,C,R,AF,ERMS,SIG,T,DET,IC)
C             USES FNEQS AND MATINV SUBROUTINES.(MUST INCLUDE CARDS.)
C (I) X       IS A VECTOR OF INDEPENDENT VARIABLE.
C (I) F       IS A VECTOR OF FUNCTION VALUES FOR POLYNOMIALS.
C (I) M       IS NUMBER OF POLYNOMIALS
C (T) A       IS THE MATRIX OF AT LEAST (N+1)X(N+2),IS REPLACED BY INVERSE
C (I) NRA     IS NUMBER OF ROWS IN A
C (I) N       IS DEGREE OF POLYNOMIAL, N.LE. 99
C (R) C       IS VECTOR FOR (N+1)COEFFICIENTS
C (R) R       IS VECTOR FOR M RESIDUALS
C (R) AF      IS VECTOR FOR M APPROXIMATE FUNCTIONS
C (R) ERMS    IS THE ROOT MEAN SQUARE ERROR,ZERO IF M.LE.N+1
C (R) SIG     IS A VECTOR FOR N+1 SIGMAS.
C               SIG IS INVERSE ELEMENT IF INV. ELEMENT IS NEGATIVE.
C (R) T       IS A VECTOR FOR N+1 T VALUES.
C (R) DET     IS THE VALUE OF THE DETERMINANT.
C (I) IC      IS THE CONTROL--
C         IC IS 0   COMPUTE EVERYTHING.
C         IC IS 1   COMPUTE ONLY COEFFICIENTS.
C         IC IS 2   COMPUTE EVERYTHING EXCEPT RESIDUALS AND APPROXIMATIONS.
C         IC IS 3   COMPUTE EVERYTHING EXCEPT APPROXIMATIONS.

      SUBROUTINE FNMIN(N,X,FX,FUN,E,EPS,K)
C             FINDS MINIMUM OF A FUNCTION OF MORE THAN ONE VARIABLE.
C (I) N       IS THE NUMBER OF VARIABLES. N<11 UNLESS CHANGE DIMENSION STATS.
C(IR) X       IS A LINEAR ARRAY CONTAINING THE INITIAL ESTIMATES OF THE N
C             VARIABLES AND AT RETURN CONTAIN THE VALUES AT THE MINIMUM.
C (R) FX      IS WHERE THE FUNCTIONAL VALUE AT THE MINIMUM WILL BE STORED.
C (F) FUN     IS THE NAME OF A FUNCTION OF 2 ARGUMENTS--FUN(X,N)--THAT
C             COMPUTES THE VALUE OF THE FUNCTION AT X. (AN EXTERNAL
C             STATEMENT IN THE CALLING PROGRAM IS NECESSARY).
C (I) E       IS THE NAME OF A SCALAR WHICH IS USED TO DEFINE THE INITIAL
C               TRIAL STEP AND THE INITIAL BOUND FOR THE CHANGE IN EACH
C               VARIABLE. E>1. (DELX(I)) INITIAL=E*EPS(I) AND
C               (DELX(I))MAX.  INITIAL=20*(E*EPS(I)).
C (I) EPS     IS A LINEAR ARRAY OF N EPSILONS DEFINING THE ACCURACY
C               DESIRED IN EACH OF THE VARIABLES.
C(IR) K       IF K=0 INITIALLY, AN ERROR PRINT AND HALT WILL BE
C               EXECUTED WHENEVER CONVERGENCE WITHIN EPS HAS NOT BEEN
C               ACHIEVED AFTER 20*N ITERATIONS.IF K IS NOT ZERO INITIALLY,
C               RETURN IS EXECUTED UPON CONVERGENCE WITH K SET TO 1,
C               OR AFTER 20*N ITERATIONS WITH K SET TO 2.
```

```
      SUBROUTINE FDMIN(N,X,DX,F,SUB,D,EPS,EPSI,K)
C             FINDS MINIMUM OF A FUNCTION, USES DERIVATIVES.
C               MUST BE FUNCTION OF MORE THAN ONE VARIABLE, I.E. N.GT.1.
C (I) N       IS THE NUMBER OF INDEPENDENT VARIABLES IN THE FUNCTION TO
C               BE MINIMIZED.N<11 UNLESS DIMENSION STATEMENTS ARE MODIFIED.
C(IR) X       IS THE LINEAR ARRAY OF VARIABLES.INITIALLY CONTAIN THE
C               ESTIMATES OF THE VALUES AT THE MINIMUM.AT RETURN
C               CONTAIN THE FINAL VALUES.
C (T) DX      IS A LINEAR ARRAY CONTAINING THE VALUES OF THE N PARTIAL
C               DERIVATIVES OF THE FUNCTION EVALUATED AT X BY THE SUB
C               PROGRAM.NO INITIAL VALUES REQUIRED.
C (R) F       CONTAINS THE VALUE OF THE FUNCTION AT RETURN.
C (S) SUB     IS THE NAME OF A SUBROUTINE--SUB(N,X,F,DX)--THAT COMPUTES
C               THE FUNCTIONAL VALUE (F) AND DERIVATIVES (DX).
C (I) D       IS AN ESTIMATE OF THE IMPROVEMENT IN THE VALUE OF THE FUNCTION.
C               WHEN D=0, ROUTINE ASSUMES THE MIN. VALUE IS NEAR 0.
C (I) EPS     IS THE ACCURACY DESIRED IN THE FUNCTION VALUE.
C (I) EPSI    IS A CONDITION ON THE INDEPENDENT VARIABLES.
C               ABS(DELTAX(I))/ABS(X(I))<EPSI.IGNORED IF EPSI VALUE=0.
C(IR) K       IF K IS INITIALLY ZERO, AN ERROR PRINT AND STOP WILL BE
C               EXECUTED WHEN FUNCTION IS NOT CONVERGING.IF K IS NOT ZERO
C               INITIALLY,RETURN IS EXECUTED WITH K SET TO 1 WHEN
C               CONVERGENCE IS SATISFIED OR K SET TO 2 WHEN THERE IS NOT
C               CONVERGENCE.
```

Standard FORAST Subroutines

Most of these descriptions are taken from BRL Report No. 1273, The FORAST Programming Language for ORDVAC and BRLESC (Revised), by L. W. Campbell and G. A. Beck, March 1965. The descriptions of FN.MIN and FD.MIN are unpublished documents available from Systems Programming.

D.D.IN)X)FX)Xo)Fo)tpt)n)ix) if)%

(Divided Difference Interpolation)

Must use all three optional arguments or none. If omitted, (5)1)1) is used.

D.D.SX)Fo)FX)%

Use this to interpolate more functions using the

X is the address of the argument.

FX is the address of the result.

Xo is the initial address of the table of Xi's.

Fo is the initial address of the table of Fi's.

tpt is the number of entries in the table. (no. of Xi's)

n is the number of points to use in the interpolation.

ix is the distance between entries

same value of X. (must use ENTER).

in the X table if is the distance between entries in the F table.

The following matrix manipulation routines are available:

S.N.E	)A1,1)n)Co)DET%	A1,1: B1,1: C1,1 are addresses of the first elements of matrices.
MAT.INV.	)A1,1)n)Co)DET%	
	To omit the use of Co and use DET it is necessary to write A1,1)n))DET)	n is the number of unknowns (rows).
		Co is the address of the first element of the solution.
		DET is the address of the determinant.
SY.SNE	)A1,1)n)Co)DET%	C1 is the address of the first
SY.INV	)A1,1)n)Co)DET%	CAP coefficient of the given equation.
F.N.E.	)A1,1)n)C1)W%	W is the address at weights for the equation.
F.O.MAT	)A1,1)n)%	
MAT.M	)A1,1)B1,1)C1,1)i) j)k)%	i is the number of rows in A(or A^T).
		j is the number of cols. in A(or A^T) and is equal to the no. of rows in B(or B^T).
		k is the number of columns in B(or B^T).

Additional comments on the above matrix subroutines: The S.N.E. (Solve normal equations) assumes all elements of a matrix having n rows and n + 1 columns are stored in the memory by rows. The SY.SNE (symmetric solve normal equations) assumes that only the upper triangle of an n x n + 1 matrix is stored and SY.INV (symmetric inversion) assumes that only the upper triangle of an n x n matrix is stored. S.N.E.; MAT.INV; SY.SNE; and SY.INV all replace the original matrix with its inverse. The SY.SNE stores the solution vector only at Co. The F.N.E. (form normal equations) assumes that the upper triangular augmented matrix has been cleared by the program before it is entered with the first equation. The F.N.E. produces a matrix that can be solved with the SY.SNE. The F.O. MAT (fill out matrix) will take an

augmented upper triangular matrix (as generated by F.N.E.) and replace it with an augmented square matrix (as needed by S.N.E.).

The S.N.E. will attempt to rearrange rows of the matrix when it finds a zero diagonal element while it is computing the inverse. The row rearrangement does not affect the arrangement of the solution vector, however the inverse matrix will not be correct if any rows were actually rearranged. Rearrangement can be avoided by use of the "not" option as explained below.

Additional BRLESC S.N.E. options:

S.N.E.)A1,1)n)Co)DET)drow)dcol)B1)db)dc)ZERO)not)%

If "drow" is specified, it is the spacing between rows; i.e. the address A2,1 - address A1,1.

If "dcol" is specified, it is the spacing between columns (which is the same as spacing between elements within a row).

If B1 is specified, the n positions beginning at B1 are used as the column vector instead of the (n+1) column of the matrix.

If "db" is specified, it is the spacing between the elements of the column vector.

If "dc" is specified, it is the spacing between elements of the solution vector.

If ZERO is specified, it is the address of the number which will be used to check for zero diagonal elements. Those diagonal elements whose absolute value are less than ZERO will be considered as zero for the rearrangement test.

If "not" is any address different from zero, the S.N.E. will not rearrange any rows.

When any or all of these spacing options are omitted (or zero), the normal consecutive spacing of elements is assumed.

For MAT.INV "drow", "dcol", ZERO, and "not" may be specified when needed and have the same meaning as for the S.N.E. except "not" has the opposite meaning. MAT.INV does not normally rearrange any rosw and will do so only when "not" is specified as non-zero.

Note that when optional addresses are omitted any place except at the end of an ENTER statement, the right parenthesis must still be written for each omitted address. In particular, the above options for the MAT.INV subroutine must correspond to the same position on the list of addresses as used by the S.N.E. since they are just different entrance points to the same subroutine.

G.L.SQ

or

P.L.SQ)X)ix)F)if)m)A1,1)n)C)R)ir)AF)iaf)ERMS)SIG)T)DET)w)iw)EQSEQ)TSEQ %

(General or polynomial least squares data fitting.)

X For G.L.SQ, X is the location of the first term of the first equation. Terms must be stored consecutively.
For P.L.SQ, X is the first independent variable.

ix For G.L.SQ, ix is the distance from one equation to the next one. For P.L.SQ, ix is the distance from one independent variable X to the next one.

F is the function value for the first equation or polynomial.

if is the distance between function values.

m is the actual total number of equations or "points" that are to be used in computing the fit. (It must not include those skipped by using EQSEQ.)

A1,1 is a block of storage that must be large enough for an augmented (n x n) symmetric matrix.

n For G.L.SQ, n is the actual number of terms to be used in each equation. (It must not include those skipped by using TSEQ.) For P.L.SQ, n is one less than the number of terms and is the degree of the polynomial when all the terms are used.

C is the initial address for consecutively storing the n coefficients. (If $n \geq 38$, n + 1 spaces must be allowed at C.)

R is the initial address for storing the m residuals.

ir is the distance desired between residuals, i.e. the increment for the R address.

AF is the initial address for storing the m approximate function values.

iaf is the increment for the AF address.

ERMS is the store address for the root-mean-square error. Zero is stored when $m \leq n$ or when $\sum W_i \leq n$.

SIG is the initial address for consecutively storing the n "sigmas".

$$SIG_i = ERMS * SQRT(inv.el.A_{i,i})$$

(If the inverse element $A_{i,i}$ is negative, it is stored for SIG_i and $T_i = 0.$)

T is the initial address for consecutively storing the n "t's".

$$T_i = C_i/SIG_i$$

DET is the address to store the determinant.

W is the initial address of the weights to be used.

iw is the increment for the W address.

EQSEQ is the initial address of a consecutive sequence of numbers that have a one to one correspondence with each equation (or point) stored at X. A zero number indicates that the corresponding equation (or point) is to be used and a non-zero number indicates that it should not be used. Note that this sequence, if used, must contain m zero numbers.

TSEQ is the initial address of a consecutive sequence of numbers that have a one to one correspondence with the terms in each general equation or with the powers of X in a polynomial. A zero number indicates that the corresponding term or power of X should be used and a non-zero number indicates that it should not be used. Note that this sequence, if used, must contain n zero numbers.

A FUNCTION MINIMIZING SUBROUTINE WITHOUT CALCULATING DERIVATIVES

Donald Taylor and John Wortman Jan 1966

The subroutine will find the minimum of a function $f(X_1....X_n)$. It requires that the programmer provide for the evaluations of the function. (At present, available on BRLESC only).

The entrance sequence is as follows:

ENTER(FN.MIN)n)X_1)F)EVF)E)H_1)EPS_1)ERR)

n is the number of independent variables in the function to be minimized.

X_1 is the address of the first of n consecutive positions for the independent variables. Initially, these positions should contain estimates of the values at the minimum; upon exit from the subroutine, they contain the final values of the independent variables.

F is the address of the value of the function.

EVF is the address of the Programmer's function evaluation. It must use $X_1....X_n$ as the values of the independent variables and store the resulting function value at F. Use the name (FN.RET) to return to the subroutine.

E is the address of a number which is used to define the initial trial step and the initial bound for the change in each independent variable. Normally, $E > 1$.

$(\Delta X_i)initial = E \times EPS_i$ $(\Delta X_i)max.initial = 20(E \times EPS_i)$

H_1 is the address of the first of $n^2 + 4n$ consecutive positions used for temporary storage. (Direction matrix, etc.)

EPS_1 is the address of the first of n consecutive positions which should contain the accuracy desired in each of the independent variables.

ERR OPTIONAL. The subroutine will send control to this address (instead of producing an ERROR print) if, after 20n iterations, the requested accuracy has not been satisfied.

It is possible to maximize a function f by minimizing the function -f. In least squares fitting, the subroutine has been used successfully to fit some non-linear functions for which 'differential corrections' did not work well. In this case, the function to minimize is the sum of the squares of the residuals.

The subroutine is derived from a method described by M.J.D.Powell.[1] It is a variation of the well known method of minimizing a function of several variables by changing one parameter at a time.

The method does not recognize constraints on the variables. Sometimes, it may be possible to apply some constraints within the function evaluation program by assigning some relatively large value to the function whenever the constraint has been violated. There is, however, no assurance that this will succeed.

The error print when the number of iterations exceeds 20 n (unless bypassed by the option) is ITER.>20xN and the number printed is the minimum value of the function at this point.

It is suggested that several sets of initial values of the independent variables should be tried to see that they converge to the same minimum to give some assurance to the result. The subroutine does not do this.

A FUNCTION MINIMIZING SUBROUTINE (WITH DERIVATIVES)

J.C. Torrey and John Wortman November 1965

This subroutine will find the minimum of a function $f(x_1 \ldots x_n)$. The routine is called FD.MIN to emphasize that the programmer must provide evaluations for the partial derivatives of his function as well as for the function value. At present it is available only on BRLESC.

The entrance sequence is as follows:

ENTER(FD.MIN)n)X1)EVF)D)EPS)H1)F)EPSI)%, where

n is the number of independent variables in the function to be minimized.

[1] M.J.D.Powell; An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives. The Computer Journal, Vol. 7, No. 2, July, 1964.

X1 is the address of the first of n consecutive positions for the independent variables. Initially, these positions should contain the programmer's estimates of the values at the minimum; when the routine is finished, they hold the final values of the independent variables.

EVF is the address of the programmer's function and derivative evaluation. Return from EVF to the routine is done by GOTO(FD.RET)%.

D is the address of an estimate of the improvement in the value of the function. This estimate will not be critical except in cases where the routine is to distinguish between local minima. (Option: When D = 0, the routine assumes that the function's minimum value is near zero and ≥ zero.)

EPS is the address of the accuracy desired in the function value.

H1 is the address of the first of $n^2 + 9n$ consecutive positions, used for temporary storage.

F is the address of the first of n+1 positions for the value of the function and its n partial derivatives. (Note that the function is stored at F, with derivatives following in the same order as the corresponding variables at X1.)

EPS1 (optional) is the address of a cauchy-like condition on the independent variables. At the last routine iteration,

$$\frac{|\Delta X_i|}{|X_i|} < \text{EPS1}, \qquad i=1...n$$

The programmer will use the EPS1 option when his interest is in the values of his variables at the minimum, rather than the function itself.

Note that FD.MIN can be used to maximize a function. To maximize f, minimize -f.

The routine has been used successfully in least squares fitting that would normally be done by differential corrections. To do this, the function to minimize is the sum of the squares of the residuals. The value of D at entrance should be zero.

FD.MIN, derived from a routine described by Davidon[1]; uses a variable n x n matrix M as a metric in searching for the best values of the X_i. The programmer desiring to impose constraints on his variables, or to use information about them to speed the search, should modify M the first (but only the first) time the routine enters his function evaluation. M is stored in the first n^2 positions of the block H1, and is set to the unit matrix at the start by the subroutine.

Setting the diagonal elements of M to the squares of the estimated error in the initial values of the variables may effect a substantial increase in the speed of minimization. Thus, if $X_3 = 14 \pm 4$, then $m_{3,3} = 16$; but if $X_3 = 14 \pm .1$, then $m_{3,3} = .01$.

If the programmer desires to impose linear constraints on his variables, he modifies the matrix at the first function evaluation. For the constraints

$$\sum a_i x_i = k_1 ,$$

$$\sum b_i x_i = k_2 , \text{ etc.},$$

he must choose M so that

$$\sum a_i m_{ij} = 0$$

$$\sum b_i m_{ij} = 0 , \text{ etc.},$$

and initial values of the X_i to satisfy the constraints.

The subroutine has an error exit which prints 'FDMIN DOWN' for its error number. A faulty derivative evaluation is the most likely cause of error, but the programmer should assure himself that his function has a minimum.

Finding a minimum of a general function is an uncertain process. Davidon suggests converging to the minimum several times from varying initial values before accepting it. FD.MIN does not do this, but its users may add assurance to their results by following his lead.

---

[1] William C. Davidon, "Variable Metric Method for Minimization", ANL-5990, Physics and Mathematics (TID-4500, 14th ed.) AEC R&D Report.

FORAST Programs

The LEAST SQUARES PROGRAM has been extended considerably since its original description and amendment (Reference 12) were written in 1965. This program is still available, but the Computer Support Division recommends using the MULTIPLE REGRESSION program instead.

The description of the MULTIPLE REGRESSION program (Reference 13) is much to bulky to include here. The description of the NONLINEAR LEAST SQUARES PROGRAM is included. This description refers to Reference 13.

FORTRAN programs and descriptions for both MULTIPLE REGRESSION and NONLINEAR LEAST SQUARES will be available by the time this report is published.

The LEAST SQUARES CUBIC SPLINE is the only other description available at this time for programs in this section.

The PIECEWISE QUARTIC FIT, and the CUBIC SPLINE have been used by various members of the Firing Tables Branch of EBL. They may now have programs and instructions for them as they have for the LEAST SQUARES CUBIC SPLINE.

NONLINEAR LEAST SQUARES PROGRAM

L. Campbell December 1966

The multiple regression computer program (as described in BRL Report No. 1330) has been used as a basis for developing a new general purpose non-linear least squares program for BRLESC. Except for the exceptions noted here, the rules for using this program are the same as for the regression program.

For the non-linear least squares, the expression that is to be used to fit the observed data must be included as a redefinition formula except it must be called F. The right side of this formula must be inclosed in parentheses if it has + signs between "terms".

For example:

$$F = (A + EXP(A1*V1)) ,$$

where A,A1,...,A24 must be used to refer to the coefficients that are to be determined from initial approximations to these coefficients. (A and Ao both refer to the first coefficient.)

The partial derivatives $\frac{\partial F}{\partial A}$, $\frac{\partial F}{\partial A1}$,..... must also be defined by redefinition formulas and they may be called P,P1,P2,... (or R's may be used if R's are also used in the main formula). For example, the partial derivatives of the above F expression would be written as:

$$P = 1, \; P1 = V1 * EXP(A1*V1) ,$$

The main formula must define a residual as the sum of all the partial derivatives, so on the left of the = symbol should be "some V(or R) - F" and on the right, each i-th term is the name used for the partial derivative with respect to the i-th coefficient. The main formula for the above example and using V4 as the observed value of the function would be

$$V4 - F = P + P1\%$$

Following the main formula must be one or more lines of initial estimates for the coefficients punched in 10 column fields with eight per line. They should be punched with decimal points and may have exponents.

Following the coefficient estimates must be epsilons for testing convergence of the coefficients. Each coefficient must have its own epsilon and they are to be punched in 10 column fields with eight per line and may have decimal points and exponents.

The data, with optional header lines, follows and it must be followed with 2 blank lines. Data may be on tape or cards or previous data may be used by using just 2 blank lines, just like the regression program. The initial value of AH is 1 in this program too, so a single header card is expected to precede the data. The amount of data allowed on tape is unlimited, the amount of data allowed on cards is limited to the BRLESC memory capacity - 9000 approximately.

The program limits the number of iterations to 20 unless an "MI = i" control card is used where i is a new maximum number of iterations. When convergence is not reached within this maximum number of iterations, the program prints "DIDN'T CONVERGE IN i ITERATIONS." and will not print any residuals, sigmas or t's.

An "USE DELTA* p" control card may be used to cause each change in the coefficients to be less (or more) than what the program computes. Using $p < 1$ may allow convergence when $p = 1$ (its normal value) will cause divergence. ($p > 1$ should not be used.) Each computed change in each coefficient is multiplied by p before it is used to actually change the coefficient for the next iteration.

There is no rescaling in this non-linear program. All residuals and approximate functions printed will be in their redefined scale.

The output includes for each iteration, the current root-mean-square error, the maximum residual, the coefficients before being changed, the "delta" or amount of change for each coefficient and the new coefficients.

After convergence, the program normally prints all the approximate values of the function and residuals, the same as the regression program except there is no rescaling. The final root-mean-square error and maximum residual is printed and the "sigmas" and "t's".

The following control cards may be used and have the same meaning as in the regression program.

PR.RES.>=	CARD INPUT
NO RESIDUALS	TAPE INPUT
RESIDUALS	TAPE MF TO
STOP ERMS=	TAPE MFMF
ANGLES ARE IN MILS	SAME FORMULAS
ANGLES ARE IN DEGREES	
ANGLES ARE IN RADIANS	
VW=	
VN=	
V.=	
IDE	
AH=	

New Control Cards:

MI= i	where i is maximum no. of iterations allowed. (i = 20 initially.)
USE DELTA* p	where p is multiplied times actual computed coefficient "deltas" before actually computing new coefficients.

Input Sequence:

Control Cards
Re-definition formulas (must include F = formula) or SAME FORMULAS
Main formula (Observed Function - F = Partial derivatives.)

Initial Coefficient Estimates (10 column fields)

Epsilons for convergence for each coeff. (10 column fields.)

Heater cards, if AH $\neq$ 0 — Omit to re-use same data.
Data — Data may be on tape or cards
2 blanks

(May repeat above sequence any number of times.)

Output Sequence:

Header lines, if any.
First 12 data numbers.
Two lines of program parameters.
All formulas.
blank line

Current ERMS and Max. Residual
Initial Coefficients
Computed changes of coefficients — repeated for each
If p $\neq$ 1, actual changes of coefficients — iteration.
New coefficients
blank line

Approximate function and Residuals

or

Data line no., original function, and residual.

Final ERMS and final Max. Residual

Sigmas

t's (If t = 0, then corresponding sigma is negative diag. matrix element.)

(Next output sequence will start on a new page.)

PROGRAM CUBIC SPLINE LEAST SQUARE

PROGRAMMER JOE HURFF

DATE MAY 1969

LANGUAGE FORAST

DESCRIPTION FOR GIVEN DATA SETS, THIS PROGRAM YIELDS AN APPROXIMATING FUNCTION IN THE FORM OF A SERIES OF CUBIC POLYNOMIALS. THE FUNCTIONS ARE DERIVED BY USING BOTH THE PRINCIPLE OF LINEAR LEAST SQUARES AND CUBIC SPLINE FITTING. THIS PROCESS MAY USUALLY BE USED SUCCESSFULLY FOR SMOOTHING AND PROVIDING FIRST AND SECOND DERIVATIVES.

PROCEDURE

I DATA CARDS (12 DIGIT FLOATING POINT) THERE MAY BE A MAXIMUM OF 500 DATA POINTS

1. CARD SET-UP

COLS	
1-12	X_I
13-24	$F(X_I)$
25-80	BLANK

2. BREAK POINTS (12 DIGIT FLOATING POINT) ONE OR TWO CARDS

COLS	CARD 1	CARD 2
1-12	$\bar{X}_0$	$\bar{X}_6$
13-24	$\bar{X}_1$	$\bar{X}_7$
25-36	$\bar{X}_2$	$\bar{X}_8$
37-48	$\bar{X}_3$	$\bar{X}_9$
49-60	$\bar{X}_4$	$\bar{X}_{10}$
61-72	$\bar{X}_5$	S *

* If there are fewer than 10 break points, the S must follow in field immediately after the last one.

II HEADER CARDS (NONE)

III CONTROL CARD (12 DIGIT FLOATING POINT)

COLS	
1-12	$F'(X_0)$
13-24	$F'(X_N)$
73-75	3 DIGIT CODE - USER'S OPTION
76	0 - FIT $F'(X_0)$ EXACTLY 1 - CONSTANT $F''(X)$ OVER FIRST INTERVAL 2 - FIT $F(X_0)$ EXACTLY
77	0 - FIT $F'(X_N)$ EXACTLY 1 - CONSTANT $F''(X)$ OVER THE LAST INTERVAL 2 - FIT $F(X_N)$ EXACTLY
78-80	3 DIGIT CODE - USER'S OPTION

IV ORDER OF INPUT

1. CONTROL CARD
2. BREAK POINT CARDS
3. DATA CARDS
4. BLANK
5. PROB CARD

TIME APPROXIMATELY ONE HALF MINUTE PER CASE

OUTPUT THE OUTPUT IS COMPLETELY LABELED

Other FORTRAN Routines

No description of the Share program we called CONSTRAINED NONLINEAR LEAST SQUARES is given. Dr. Celmins kindly prepared a description of the NONLINEAR LEAST SQUARES FOR CORRELATED DATA routine. The description of NLPROG was taken from BRL Memorandum report No. 1958.

NON-LINEAR LEAST SQUARES SUBROUTINES FOR CORRELATED DATA (COLSA, COLSB)

A. Celmins December 1968

Reference: (18) BRL Report 1360 (March 1967), Appendix

Problem Outline

A general least squares problem can be formulated as follows: We have observed r sets of n quantities $x_1,\ldots,x_n$. Between these quantities a functional relationship which depends on m unknown parameters $y_1,\ldots,y_m$ is assumed:

$$F(x_1,\ldots,x_n\ ;\ y_1,\ldots,y_m) = 0 \qquad (1)$$

If $r > m$, we compute the parameters and some corrections of the observations such that Equation (1) is satisfied at all r observation sets and the sum of correction squares is a minimum. For this computation the correction squares should be weighted differently depending on their accuracies and on correlations between them.

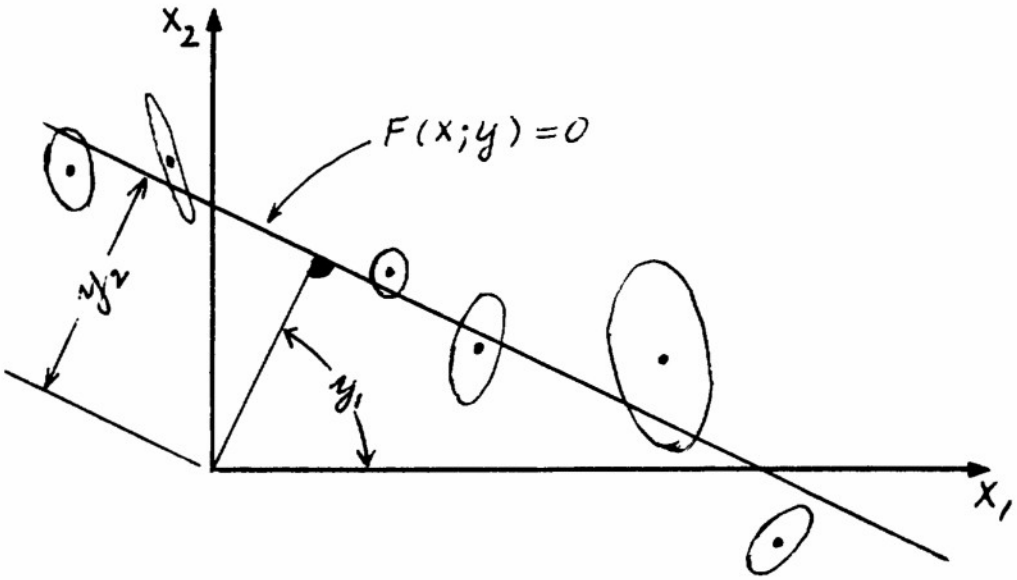
The subroutine COLSA computes the values of the parameters y_j, their accuracies, and their cofactor matrix (i.e. variance-co-variance matrix) from the following data: Observations, standard errors of the observations, a cofactor matrix for each observation set, and approximate values of the parameters. Correlations between observations belonging to different sets are not considered.

The subroutine COLSB may be called after the final values of the parameters are established. It computes the corrections (probable errors) of the observations, prints and plots error distributions, prints identifications of sets with large corrections and carries out some numerical controls of the computations.

Sample Problem

The relationship between x_1 and x_2 may be linear:

$$F(x_1,x_2\ ;\ y_1,y_2) = x_1 \cos y_1 + x_2 \sin y_1 - y_2 = 0$$



The observed points (x_1,x_2) may have different accuracies and there may be some correlation between the coordinates observed so that the error ellipses of the points observed have different sizes and directions. (Such correlations can be caused by the observation technique, for instance.) COLSA will assign different weights to the points in accordance with their error ellipses. The resulting parameters y_1 and y_2 will be furnished together with their standard errors and cofactor matrix. (y_1 and y_2 are correlated because both are computed using the same data.)

Data for COLSA and COLSB

All data are assumed to be in the core memory. The dimensions of the corresponding arrays are as follows:

Observations: X(5,1000) - $n \leq 5$; $r \leq 1000$

Standard errors of unit weight: ERZX(1000) - $r \leq 1000$

Cofactor matrices: RCOR(5,5,1000) - $n \leq 5$; $r \leq 1000$

Approximations of parameters: YCAP(10) - $m \leq 10$

Alphanumeric identifications of observed sets, consisting of two 10-letter words for each set: IDEN (1000)
TIFIC (1000) - $r \leq 1000$

In addition to the data, a subroutine, say FU, to evaluate the function $F(x_1,\ldots,x_n ; y_1,\ldots,y_m)$ is needed. The routine must furnish the value of F as well as the n + m values of the first partial derivatives $\partial F/\partial x$ and $\partial F/\partial y$. The calling of COLSA and COLSB is described below.

Calling COLSA

CALL COLSA(N,M,FU,YCAP,YLOW,ERY,ERZERO,Q,NR,X,ERZX,RCOR,NTOT,ETA,IDEN, TIFIC)

In the following list, I = input argument provided by calling routine and R = result set by COLSA.

I-N = number of observations in each set ($N \leq 5$)

I-M = number of parameters ($M \leq 10$)

I-FU = name of subroutine for $F(\bar{x},\bar{y})$

I-YCAP = approximate values of the parameters. Dimension: YCAP(10)

R-YLOW = YCAP + ETA = improved parameter values. Dimension: YLOW(10)

R-ERY = standard errors of YLOW. Dimension: ERY(10)

R-ERZERO = standard error of weight one (associated with cofactor matrix Q)

R-Q = cofactor matrix of YLOW. Dimension: Q(10,10)

I-NR = number of sets observed ($NR \leq 1000$)

I-X = observations. Dimension: X(5,1000)

I-ERZX = standard error associated with each set. Dimension: ERZX(1000)

I-RCOR = cofactor matrices of observation sets. Dimension: RCOR(5,5,1000)

R-NTOT = number of valid observation sets. Normally NTOT=NR; see description of FU - subroutine.

R-ETA = corrections of YCAP. Dimension: ETA(10)

I-IDEN, I-TIFIC = alphanumeric identifications of observation sets. Dimension: IDEN(1000), TIFIC(1000)

The results of COLSA are printed in a self-explaining way. The following symbols are used by the routines for printouts:

X = observations

X + KSI = corrected observations

Y = approximations of parameters

Y + ETA = corrected parameters

G = weight of observation set

F(X,Y + ETA) = constraint function with corresponding arguments (Equation (1)).

There is no checking by COLSA whether N or M exceeds 5 or 10, respectively. Also, the value of NR is not checked to see if it satisfies $M < NR \leq 1000$.

Programming $F(\bar{x},\bar{y})$

The constraint function is entered by COLSA and COLSB as follows CALL FU(X,Y,FVA,A,B,L,NMES) where

I-X = observations. Dimension: X(5,1000)

I-Y = parameter vector. Dimension: Y(10)

R-FVA = $F(x_1,\ldots,x_n ; y_1,\ldots y_m)$

R-A = $\partial F/\partial x$. Dimension: A(5)

R-B = $\partial F/\partial y$. Dimension: B(10)

I-L = number of set. The arguments $x_1,\ldots,x_n$ are:
$x_1 = X(1,L)$, $x_2 = X(2,L),\ldots,x_n = X(N,L)$

R-NMES = error message indicator. NMES = 1 if the function or its derivatives cannot be computed with the arguments given. If this happens, COLSA will print a comment and the identification of the corresponding set and reduce the total number of valid sets, NTOT by one.

Calling COLSB

CALL COLSB(N,M,FU,YCAP,YLOW,ERY,ERZERO,Q,NR,X,ERZX,RCOR,ETA,IDEN,TIFIC)

The arguments of COLSB are the same as those of COLSA, except NTOT is not an argument of COLSB. In contrast to COLSA, all COLSB arguments are of input type. Normally they will have the values furnished by the

last call of COLSA. All output of COLSB is on the printer (in self-explaining manner) and plotter tape. The plotted information includes the frequency distribution, cumulative histogram and probit diagram of the weights of F(X,Y + ETA). For comparison the corresponding number distributions and normal distributions are included. The printed information includes lists of sets with large errors, surveys about the corrections and their distributions and some numerical tests for accuracy and randomness.

Subroutines Used

The following names are names of subroutines used by COLSA and COLSB.

CHOLES - cholesky algorithm routine for solving normal equations

PLOCOB - plotting routine used by COLSB

PLODIS
FREDIS
CUMMIS — subroutines used by PLOCOB
PRODIA
PROBIT

ERF(X) - function routine furnishing normal distribution function (error integral)

PLTCCA
PLTCCT
PLTCCD — plotter subroutines.
PLTCCS
PLTCCB

NLPROG (Taken from BRL Memorandum Report No. 1958)

In this section instructions to use the NLPROG programs will be given. These instructions will usually be given as if POWELL were the minimizing method. The necessary modification for the other minimizing subroutines will be noted.

The problem we wish to solve is: Minimize

$$F(X) \qquad X = (x_1, x_2, \ldots, x_N)^T$$

subject to

$$R_i(X) \geq 0 \qquad i=1,2,\ldots,NIC$$

and

$$R_i(X) = 0 \qquad i=NIC+1,\ NIC+2,\ldots,NC.$$

NLPROG actually tries to minimize the function

$$P(X,\rho_k) = F(X) + \rho_k \sum_{i=1}^{NIC} 1/R_i(X) + (1/\rho_k^{\frac{1}{2}}) \sum_{i=NIC+1}^{NC} R_i^2(X)$$

for $\rho_k = \rho_1$, $\rho_2 = \rho_1/RATIO$, $\rho_3 = \rho_2/RATIO$,... until $\rho_L \sum_{i=1}^{NIC} 1/R_i(X) < \theta$.

NLPROG is really a set of subroutines the names of which should be avoided in other programming. A list of the subroutine names used in NLPROG with POWELL follows. The brief explanation with them assumes that X and the previous X used both satisfy all the inequality constraints.

NLPROG - This subroutine is the main routine for this collection of subroutines.

SUBPRO - A buffer between NLPROG and the minimizing subroutine.

POWELL - Minimizing method.

LINMIN - Univariate minimization scheme: find S for which $P(XI+S\cdot V,\rho)$ is a minimum (or improved sufficiently). This uses POFS to evaluate P.

POFS - Finds $X = XI + S\cdot V$ and uses POFX to find $P(X,\rho)$. This also keeps track of the X which minimizes $P(X,\rho)$, say XMIN, and the functions of XMIN; $R_i(XMIN)$ $(i=1,2,\ldots,NC)$, $P(XMIN,\rho)$, $G(XMIN,\rho)$ and $F(XMIN)$. If X is not in the accepted region $P(X,\rho)$ is set to 10^{150}.

POFX - Controls computation of $R_i(X)$ $(i=1,2,\ldots,NC)$; $F(X)$, $P(X,\rho)$, and

$$G(X,\rho) = P(X,\rho) - 2\rho \sum_{i=1}^{NIC} 1/R_i(X).$$

OUTPUT - Prints $F(XMIN)$, $P(XMIN,\rho)$, $G(XMIN,\rho)$, ρ, $(P-G)/2$, XMIN, and $R_j(XMIN)$ $(j=1,2,\ldots,NC)$.

In addition, the user must code a subroutine, EVAL(I), which puts F(X) in F if I = 0 and $R_I(X)$ in R(I) if I > 0. If we use HOOKE instead of

POWELL, LINMIN is dropped from the package. If FLETCH is used in place of POWELL the subroutine DPBYDX is added and the coder must add the subroutine DERIV(I) where if I = 0, DRDX(j) = $\partial F(X)/\partial x_j$ (j=1,2,...,N) and if I > 0, DRDX(j) = $\partial R_I(X)/\partial x_j$ (j=1,2,...,N). If NEWTON is used in place of POWELL, the DERIV(I) subroutine must also compute second derivatives: If I = 0, DDR(j,k) = $\partial^2 F(X)/\partial x_j \partial x_k$ (j=1,2,...,N; k=1,2,...,N). If I > 0, DDR(j,k) = $\partial^2 R_I(X)/\partial x_j \partial x_k$ (j=1,2,...,N; k=1,2,...,N).

The original plan was to code up different minimization schemes and change only SUBPRO to use them. Actually the main control subroutine NLPROG is slightly different for each of the four minimization schemes and the LINMIN for POWELL is different than that used with FLETCH and NEWTON.

Blank COMMON is used to identify all the numbers which are used, supplied, or needed by the MAIN program, EVAL(I), or DERIV(I). One labeled COMMON, called SUB, is used by NLPROG. The blank common list for NEWTON is (N,NC,NIC,X(100),EPS(100),Q(100),RHO,RATIO,THETA,NREPET, NRHORP,F,R(200),IPROP,DRDX(100),DDR(100,100)).

N - number of variables. N ≤ 100.

NC - number of constraints. 0 ≤ NC ≤ 200.

NIC - number of inequality constraints. NIC ≤ NC ≤ 200.

X(100) - variables. An initial guess must be put here. The current value of X is here at other times.

EPS(100) - acceptable absolute error in variables. NLPROG may not attain this accuracy. It does check that a change of EPS(i) in any one x(i) will not reduce the final P(X,ρ). EPS(i) > 0.

Q(100) - initial search steps (not too critical). The approximate errors in the initial x_i are recommended.

RHO - ρ. control parameter. ρ > 0.

RATIO - $\rho_{i+1} = \rho_i/\text{RATIO}$. RATIO > 1 (Recommend 4 ≤ RATIO ≤ 16.)

THETA - θ. final convergence parameter.

NREPET - number of times to repeat minimization. (Optional)

NRHORP - number of ρ's to repeat for each repetition of minimization. If ρ_L is the value of ρ when $\rho \sum_i 1/R_i(X) < \theta$ the minimization is restarted with $\rho = \rho_L \cdot \text{RATIO}^{(\text{NRHORP}-1)}$. The purpose of this is to push the X which satisfies $P(X,\rho)$ away from the boundary of the region $\{X|R_i(X) > 0, i=1,2,\ldots,\text{NIC}\}$. However if there are no inequality constraints, only equality constraints, a negative or zero NRHORP would reduce ρ and force closer satisfaction of the equality constraints.

F - function to be minimized.

R(200) - constraints. (The NIC inequality constraints must be stored first.)

IPROP - controls use of OUTPUT.
- 0 - no prints (except errors)
- 1 - print initial values and the solution of each subproblem.
- 2 - Print each cycle result
- 3 - Print each linear minimum. (IPROP = 1 is recommended. This gives a fair outline of the course of the problem.)

DRDX(100)- Partial derivatives of F or R_I with respect to x_i (This array is not needed for POWELL or HOOKE.)

DDR(100,100)- Second partial derivatives of F or R_I with respect to x_i and x_j. This matrix is symmetric but must be completely filled by DERIV(I) (This array is needed only if NEWTON is used.)

The quantities starting with N or I are integers. The rest are floating point numbers. This set of blank COMMON must precede the users programs.

Before calling NLPROG, the user must store N, NC, NIC, and initial values of X(I), W(I), and EPS(I) (I=1,2,...,N). He must also store RHO, RATIO, THETA, IPROP, and if desired, NREPET and NRHORP. He must supply the subroutine EVAL(I) and, if FLETCH or NEWTON are used, the user must also supply the appropriate DERIV(I) subroutine.

When NLPROG returns control to the main program, X(I) I=1,2,...,N contains the solution X, RHO has the final value of ρ, F = F(X), and R(J) = R_J(X) J=1,2,...,NC. The numbers in N,NC,NIC,EPS(I),RATIO,THETA, NRHORP, and IPROP are not changed by NLPROG.

Comments

The need for some preliminary analysis of minimization problems cannot be overstressed. All too frequently we mechanically prepare a program and discover, while trying to code check the program, that there is an obvious solution; or worse that there is no finite solution. Further, the preliminary analysis should help select reasonable starting values. If the initial X does not satisfy the inequality constraints NLPROG will search for an X that does. However, this X may be so remote from the desired solution that the program will take a long time to reach the solution.

DISTRIBUTION LIST

No. of Copies	Organization
20	Commander Defense Documentation Center ATTN: TIPCR Cameron Station Alexandria, Virginia 22314
1	Commanding General U. S. Army Materiel Command ATTN: AMCDL Washington, D. C. 20315
1	Commanding General U. S. Army Materiel Command ATTN: AMCRD,Dr.J.V.R.Kaufman Washington, D. C. 20315
1	Commanding General U. S. Army Materiel Command ATTN: AMCRD-TE Washington, D. C. 20315
1	Commanding General U. S. Army Materiel Command ATTN: AMCRD-TP Washington, D. C. 20315
1	Commanding General U. S. Army Aviation Systems Command 12th & Spruce Streets St. Louis, Missouri 63166
1	Commanding General U. S. Army Electronics Command ATTN: AMSEL-DL Fort Monmouth, New Jersey 07703
1	Commanding General U. S. Army Missile Command ATTN: AMSMI-R Redstone Arsenal, Alabama 35809
1	Commanding General U. S. Army Tank-Automotive Command ATTN: AMSTA-CL Warren, Michigan 48090
2	Commanding Officer U. S. Army Mobility Equipment Research & Development Center ATTN: Tech Docu Ctr,Bldg 315 AMSME-RZT Fort Belvoir, Virginia 22060
1	Commanding General U. S. Army Munitions Command ATTN: AMSMU-RE Dover, New Jersey 07801
1	Commanding General U. S. Army Weapons Command ATTN: AMSWE-RE Rock Island, Illinois 61202
1	Director U. S. Army Advanced Materiel Concepts Agency Washington, D. C. 20315
1	Director U. S. Army Aeronautical Research Laboratory Moffett Naval Air Station California 94035
1	Commanding Officer U. S. Army Harry Diamond Laboratories ATTN: AMXDO-TD/002 Washington, D. C. 20438
1	Commanding Officer U. S. Army Materials & Mechanics Research Center ATTN: AMXMR-ATL Watertown, Massachusetts 02172

DISTRIBUTION LIST

No. of Copies	Organization
1	Commanding General U. S. Army Natick Laboratories ATTN: AMXRE, Dr. D. Sieling Natick, Massachusetts 01762
1	Office of Vice Chief of Staff ATTN: CSAV-W-TIS Department of the Army Washington, D. C 20310
3	Commander U. S. Naval Air Systems Command ATTN: AIR-604 Washington, D. C. 20360
3	Commander U. S. Naval Ordnance Systems Command ATTN: ORD-9132 Washington, D. C. 20360

Aberdeen Proving Ground

Ch, Tech Lib
Marine Corps Ln Ofc
CDC Ln Ofc

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) U.S. Army Aberdeen Research and Development Center Ballistic Research Laboratories Aberdeen Proving Ground, Maryland	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

FUNCTION APPROXIMATION ROUTINES FOR BRLESC - A SURVEY

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

5. AUTHOR(S) (First name, middle initial, last name)

John D. Wortman

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
August 1970	72	19

8a. CONTRACT OR GRANT NO. b. PROJECT NO. RDT&E 1T061102A14B c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) BRL Memorandum Report No. 2053 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	U.S. Army Materiel Command Washington, D.C.

13. ABSTRACT

This report is a survey of the function approximation routines available for the BRLESC computer at Aberdeen Proving Ground as of April 1969. It includes a description of each routine and some general discussion.

DD FORM 1 NOV 68 1473 REPLACES DD FORM 1473, 1 JAN 64, WHICH IS OBSOLETE FOR ARMY USE.

Unclassified

Security Classification

Unclassified

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Minimization						
Interpolation						
Curve Fitting						
Function Approximation						
Least Squares						
BRLESC						
FORTRAN Routines						
FORAST Routines						

Unclassified

Security Classification